GARY
DUDLEY SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN AUTOPILOT DESIGN FOR THE
UNITED STATES MARINE CORPS'
AIRBORNE REMOTELY OPERATED DEVICE

by

Scot D. Lloyd

September 1987

Thesis Advisor Harold A. Titus

Approved for public release; distribution is unlimited.

T234286

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) 62
7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
AN AUTOPILOT DESIGN FOR THE UNITED STATES MARINE CORPS' AIRBORNE REMOTELY OPERATED DEVICE

PERSONAL AUTHOR(S)
LLOYD, Scot D.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 September	166

SUPPLEMENTARY NOTATION

COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Remotely operated, remotely piloted vehicle, RPV, optimal control, Riccati, control systems

ABSTRACT (Continue on reverse if necessary and identify by block number)

An autopilot for the U.S. Marine Corps' ducted fan hovercraft is designed using optimal control theory. Single input controllers are designed to govern the vehicle's roll rate and altitude rate. The gyroscopic coupling between the vehicle's pitch and yaw dynamics is examined and a multi-input controller is designed. A computer program called OPTCON is developed to generate optimal feedback control gains by solving the discrete matrix Riccati equation. This program is for use on portable or home IBM compatible computers. Graphic plotting of the time-varying gains and of the system time response is available for both monitor and hardcopy output.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Harold A. TITUS	22b TELEPHONE (Include Area Code) (408) 646-2560
	22c OFFICE SYMBOL 62Ts

FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

An Autopilot Design for the United States Marine Corps' Airborne Remotely Operated Device

by

Scot D. Lloyd
Captain, United States Marine Corps
B.S.M.E., United States Naval Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

ABSTRACT

An autopilot for the U.S. Marine Corps' ducted fan hovercraft is designed using optimal control theory. Single input controllers are designed to govern the vehicle's roll rate and altitude rate. The gyroscopic coupling between the vehicle's pitch and yaw dynamics is examined and a multi-input controller is designed. A computer program called OPTCON is developed to generate optimal feedback control gains by solving the discrete matrix Riccati equation. This program is for use on portable or home IBM compatible computers. Graphic plotting of the time-varying gains and of the system time response is available for both monitor and hardcopy output.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 11
A. THE CONTROL SYSTEM DESIGN PROCESS 11
B. AROD 14
C. THE PROBLEM 16
1. Gyroscopic Coupling 17
2. Multiple Control Loops 21

II. OPTIMAL CONTROL THEORY 23
A. FEEDBACK CONTROL 23
1. Why Use Feedback ? 23
2. System Classification 23
3. System Structure with Feedback 25
B. SYSTEM DEFINITION 26
1. Continuous Time Systems 26
2. Discrete Time Systems 28
3. Constraints 32
C. THE PERFORMANCE MEASURE 32
1. Quadratic Cost Function 32
2. Regulators and Trackers 32
3. Performance Weighting Factors 33
4. Specific Types of Problems 36
D. CALCULATION OF OPTIMAL FEEDBACK GAINS 37

III. CONTROL SYSTEM DESIGN FOR AROD 39
A. OVERVIEW 39
B. AROD ROLL RATE CONTROLLER 41
1. The Roll System 41
2. Roll Rate Controller Design 44
C. AROD ALTITUDE RATE CONTROLLER 73

1. The Altitude Rate System 73
2. Altitude Rate Controller Design 76
D. AROD PITCH ANGLE AND YAW ANGLE CONTROLLER 80
1. The Pitch and Yaw System 80
2. Pitch Angle and Yaw Angle Controller Design 84

IV. CONCLUSIONS AND RECOMMENDATIONS 100
A. CONCLUSIONS 100
B. RECOMMENDATIONS FOR FURTHER WORK 100

APPENDIX A: THE OPTCON PROGRAM 102
1. OVERVIEW 102
2. AN EXAMPLE PROBLEM 103
a. The Second Order Integrator 103
b. Controllability and Reachability 106
c. Observability 107
d. Solution Using OPTCON 108

APPENDIX B: OPTCON MAIN PROGRAM LISTING 134

APPENDIX C: OPTCON SUBROUTINE LISTINGS 154

APPENDIX D: PLOT88 GRAPHICS SUBROUTINE LISTING 161

LIST OF REFERENCES 164

INITIAL DISTRIBUTION LIST 165

LIST OF TABLES

1. STATE SPACE DEFINITIONS FOR CONTINUOUS TIME SYSTEMS 27
2. STATE SPACE DEFINITIONS FOR DISCRETE TIME SYSTEMS 30
3. TYPICAL COST FUNCTIONS 36
4. VARIABLE DEFINITIONS FOR AROD ROLL RATE EQUATIONS OF MOTION 42
5. EFFECT OF SAMPLING FREQUENCY ON ROLL RATE SYSTEM 46
6. ROLL RATE PARAMETERS FOR GROUP 1 49
7. ROLL RATE PARAMETERS FOR GROUP 2 54
8. ROLL RATE PARAMETERS FOR GROUP 3 59
9. ROLL RATE PARAMETERS FOR GROUP 4 64
10. VARIABLE DEFINITIONS FOR AROD ALTITUDE RATE EQUATIONS OF MOTION 73
11. INITIAL ALTITUDE RATE PARAMETERS 77
12. FINAL ALTITUDE RATE PARAMETERS 77
13. VARIABLE DEFINITIONS FOR AROD PITCH AND YAW EQUATIONS OF MOTION 80
14. INITIAL UNCOUPLED PITCH OR YAW PARAMETERS 87
15. FINAL UNCOUPLED PITCH OR YAW PARAMETERS 87
16. PITCH/YAW CONTROLLER DESIGN SCHEMES 91

LIST OF FIGURES

1.1 Schematic Drawing of AROD 15
1.2 AROD Control Vanes 16
1.3 Spinning Rotor Orientation 17
1.4 Spinning Rotor with a Force Couple Applied 19
1.5 Resultant Angular Momentum With an Applied Torque 20
2.1 Basic Control System 25
2.2 Continuous Time System 28
2.3 Time Invariant Discrete Time System 31
2.4 Comparison Between Regulator and Tracker System Structure 34
3.1 AROD Body-Fixed Coordinate System 40
3.2 Signal Flow Diagram for Roll Rate Control 43
3.3 Open Loop Bode Diagram For Roll Rate System 45
3.4 Roll Rate Time Response Using Baseline Cost Function 47
3.5 Group 1 Time Response Parameters 50
3.6 Roll Rate Time Response for Group 1 Run Number 4 51
3.7 Roll Rate Time Response for Group 1 Run Number 12 52
3.8 Roll Rate Time Response for Group 1 Run Number 15 53
3.9 Group 2 Time Response Parameters 55
3.10 Roll Rate Time Response for Group 2 Run Number 4 56
3.11 Roll Rate Time Response for Group 2 Run Number 11 57
3.12 Roll Rate Time Response for Group 2 Run Number 14 58
3.13 Group 3 Time Response Parameters 60
3.14 Roll Rate Time Response for Group 3 Run Number 4 61
3.15 Roll Rate Time Response for Group 3 Run Number 14 62
3.16 Roll Rate Time Response for Group 3 Run Number 17 63
3.17 Group 4 Time Response Parameters 65
3.18 Roll Rate Time Response for Group 4 Run Number 4 66
3.19 Roll Rate Time Response for Group 4 Run Number 10 67

3.20 Roll Rate Time Response for Group 4 Run Number 13 68
3.21 Percent Overshoot Curves For Groups 1, 2, 3, and 4 71
3.22 Settling Time Curves For Groups 1, 2, 3, and 4 72
3.23 Signal Flow Diagram for Altitude Rate Control 75
3.24 Initial Altitude Rate Time Response 78
3.25 Final Altitude Rate Time Response 79
3.26 Signal Flow Diagram for Pitch and Yaw Angle Control 82
3.27 Initial Uncoupled Pitch or Yaw Angle Time Response 88
3.28 Final Uncoupled Pitch or Yaw Angle Time Response 89
3.29 Pitch / Yaw Angle Time Response for Controller Number 1 92
3.30 Pitch / Yaw Angle Time Response for Controller Number 2 93
3.31 Pitch / Yaw Angle Time Response for Controller Number 3 94
3.32 Pitch / Yaw Angle Time Response for Controller Number 4 95
3.33 Pitch / Yaw Angle Time Response for Controller Number 5 96
3.34 Pitch / Yaw Angle Time Response for Controller Number 6 97
3.35 Pitch / Yaw Angle Time Response for Controller Number 7 98
A.1 OPTCON Program Flow Diagram 104
A.2 Second Order Integrator Signal Flow Diagram 105
A.3a Gain Trajectory for State x_1 Second Order Integrator Example 121
A.3b Gain Trajectory for State x_2 Second Order Integrator Example 122
A.4 Phase Plane for Second Order Integrator Example 127
A.5a State x_1 Time Response for Second Order Integrator 129
A.5b State x_2 Time Response for Second Order Integrator 130

ACKNOWLEDGEMENTS

I offer my sincere thanks to Professor Hal Titus for sharing his insight, wisdom, and sense of humor with me during the course of this work. Professor Titus has taught me that the pursuit of knowledge is a joyful and rewarding experience. It is with deepest gratitude that I thank my loving wife, Dawn, for her unfailing support and understanding over the past two and a half years. When the going got tough, she always managed to come through with a smile and words of encouragement. No acknowledgement is complete without recognizing our God, the creator of all knowledge and truth, who has been my strength all along the way. This thesis is dedicated to Dawn and to our son David.

I. INTRODUCTION

A. THE CONTROL SYSTEM DESIGN PROCESS

Since the beginning of time, man has sought ways to control the laws of nature. From the simple float regulator developed by the Greeks in 300 B.C. [Ref. 1: p 3], to the amazingly complex space shuttle of the 1980's, control systems span the range of mankind's efforts to govern his surroundings. The challenge for a control systems engineer is to use his knowledge, skill, judgment, and experience to systematically develop a solution to any of a number of different types of control problems. There is seldom only one right answer to a control problem. In general, there may be several alternate solutions to the same problem and the final product will probably be a compromise between them. It remains the responsibility of the engineer to choose the "best" solution that meets the performance criteria specified by the user. So, how does the engineer know where to begin when he is given a set of performance criteria for a system ? There is no set procedure carved in stone. There are, however, a few broad guidelines that give the engineer a rough idea of the tasks which need to be accomplished in his quest to design an effective control system. These milestones are as follows :

1. Define the system.
2. Specify the desired performance of the system.
3. Identify the constraints under which the system must operate.
4. Translate the information from milestones 1, 2, and 3 into a mathematical model that can be simulated on the computer.
5. Use the available tools to develop a control system which satisfies the performance specifications.
6. Evaluate the control system design using computer simulations.
7. Modify the design as required to better suit the application.
8. Incorporate the control design in a prototype system to test the ability of the system to tolerate real world non-linearities and non-ideal conditions.
9. Modify and optimize the design until a satisfactory control system is realized.

The first milestone listed above is not always as simple as it appears to be. In fact, defining the limits of the system may possibly be the most difficult phase of the design. If the engineer does not expend considerable effort in the definition of the

problem which he is to solve, he might end up solving the wrong problem. The engineer needs to include enough parameters in his system to accurately model the true system without becoming overburdened computationally. This is as much an art as it is a science. The engineer will probably need to make simplifying assumptions and approximations which tend to widen the gap between the performance of the model and the performance of the real world system.

Defining the desired performance of the system may or may not be left to the discretion of the engineer. If a strict set of specifications is handed to him, then the engineer has little choice but to satisfy those specifications or be able to defend his claim that they can not be satisfied. On the other hand, there may be considerable leeway for the engineer to make sweeping changes in the control system and still satisfy the required specifications. Several tools are available to measure the performance of a control system. The classical design engineer holds fast to such measures as the gain margin, phase margin, root locations in the S plane or Z plane, and bandwidth. The advent of optimal control techniques has placed emphasis on the minimization of some cost function as a means of measuring system performance. All of these techniques have their place in the realm of control system design and it is the mark of a successful designer that he can incorporate any or all of the tools when the situation dictates.

The third milestone is an important yet often overlooked element of the design process. Constraints on the system may include any or all of the following :

1. Monetary cost
2. Admissible control inputs
    a. Saturation limits
    b. Observability of the parameters required for control
3. Physical limitations
    a. Size
    b. Weight
    c. Minimum and / or maximum velocities, accelerations, etc.
    d. Initial conditions
    e. Final conditions
    f. Sampling rate and processor speed for digitally controlled systems

The mathematical model is the link between the real world system and the design tools which the engineer has at his disposal. In general, most physical systems which need to be modelled can be represented by some set of differential equations. The

physical laws of nature lend themselves nicely to approximation by linear ordinary differential equations with constant coefficients. Non-linearities and random processes are also quite prevalent in many systems and the effects of these phenomenon can greatly complicate the engineer's effort to model a system. That is why he must have the expertise and experience to know how to make assumptions and approximations which simplify the problem at hand to a point where he can use the available design tools.

The next step is to use design tools to develop a control system which satisfies the desired performance specifications. It is at this point that two schools of thought begin to emerge. The classical school of thought focuses on such design tools as the Root Locus Plot, Nyquist Plot, Bode Diagram, and Function Minimization. The more daring school of thought centers its attention on the maximum principle of Pontryagin and the method of dynamic programming developed by Bellman. The advent of digital computers in the mid 1950's made these more powerful design tools realizable since the amount of computation required by them was prohibitive if not impossible to do by hand. In any event, the design engineer has a myriad of tools from which to choose.

Once the design of the controller is accomplished, the next step is to integrate the control system with the system model and then simulate the entire system to evaluate its performance. A very useful method to perform this evaluation is to study the time response of the output variables of the system. Such parameters as rise time, peak overshoot, and settling time are typical values to be noted. The digital computer once again is a very useful means of obtaining such information rapidly.

Even the best of control designers is not apt to hit a bullseye on his first shot. Control system design theory does not guarantee success on every try. The method of trial and error is one with which all control engineers are familiar. Modifications to the control system are inevitable.

Once the controller design is proven in simulation studies, it is time to test it out on a prototype or small scale model of the actual system. This phase of design can become costly if the designer has not thoroughly tested his controller on the computer first. In this phase of design, the non-linearities and random effects which were ignored or approximated during the modelling phase become significant factors once again. Conditions which were assumed to be ideal in the model now become non-ideal. The evaluation process begins all over again as these new disturbances change the performance of the system.

The next step is obvious. After evaluation of the controller in a real world system, the need for further modifications is again probable if not inevitable. Changes must be made until a satisfactory controller has been designed which meets the desired specifications.

B. AROD

It is the goal of this thesis to complete the first seven steps of the nine step design process discussed in the previous section. The system chosen for this endeavor is an airborne remotely piloted vehicle (RPV) called **AROD.** The acronym stands for Airborne Remotely Operated Device. The United States Marine Corps initiated work on AROD early in 1986 and is attempting to introduce the vehicle into the operational Fleet Marine Force during fiscal 1987. AROD is a slow, low-flying ducted fan vehicle powered by a vertically mounted, two cycle, two cylinder gasoline engine which drives a three-bladed propeller. See Figure 1.1.

The vehicle is 38 inches tall, 32 inches in diameter, and has a nominal weight of 85 pounds. It presently has a payload capacity limited to a miniature television camera and a canister of fiber-optic cable. The Marine Corps plans to use AROD for short range reconnaissance and over-the-hill spy in the sky surveillance. The fiber-optic cable provides two-way communication between the vehicle and the ground based operator. The uplink communication will consist of control commands while the downlink will provide real time surveillance and on-board status information.

The primary flight mode is low altitude hovering with the axis of the spinning propeller oriented perpendicular to the surface of the earth. In order to translate horizontally across the earth's surface, the entire vehicle must be tilted towards the direction of intended movement. The mechanism by which AROD is tilted for such translation consists of four control vanes located in the airflow downstream of the propeller wash. One pair of control vanes is designated as the rudder. The other pair is designated as the elevator and is oriented such that its axis of rotation is perpendicular to the axis of rotation of the rudder pair. See Figure 1.2. All four fins assume dual responsibility for aerodynamic control in that they also serve as ailerons for AROD. The control vanes are actuated by model airplane servos. These servos are limited to a maximum deflection of $\pm$ 30° and a maximum angle rate of 50°/sec. A maximum translational velocity of 30 knots (34 mph) in a no wind condition is desired. The translational velocity of AROD is proportional to the tilt angle created by the rudder and elevator control vanes.

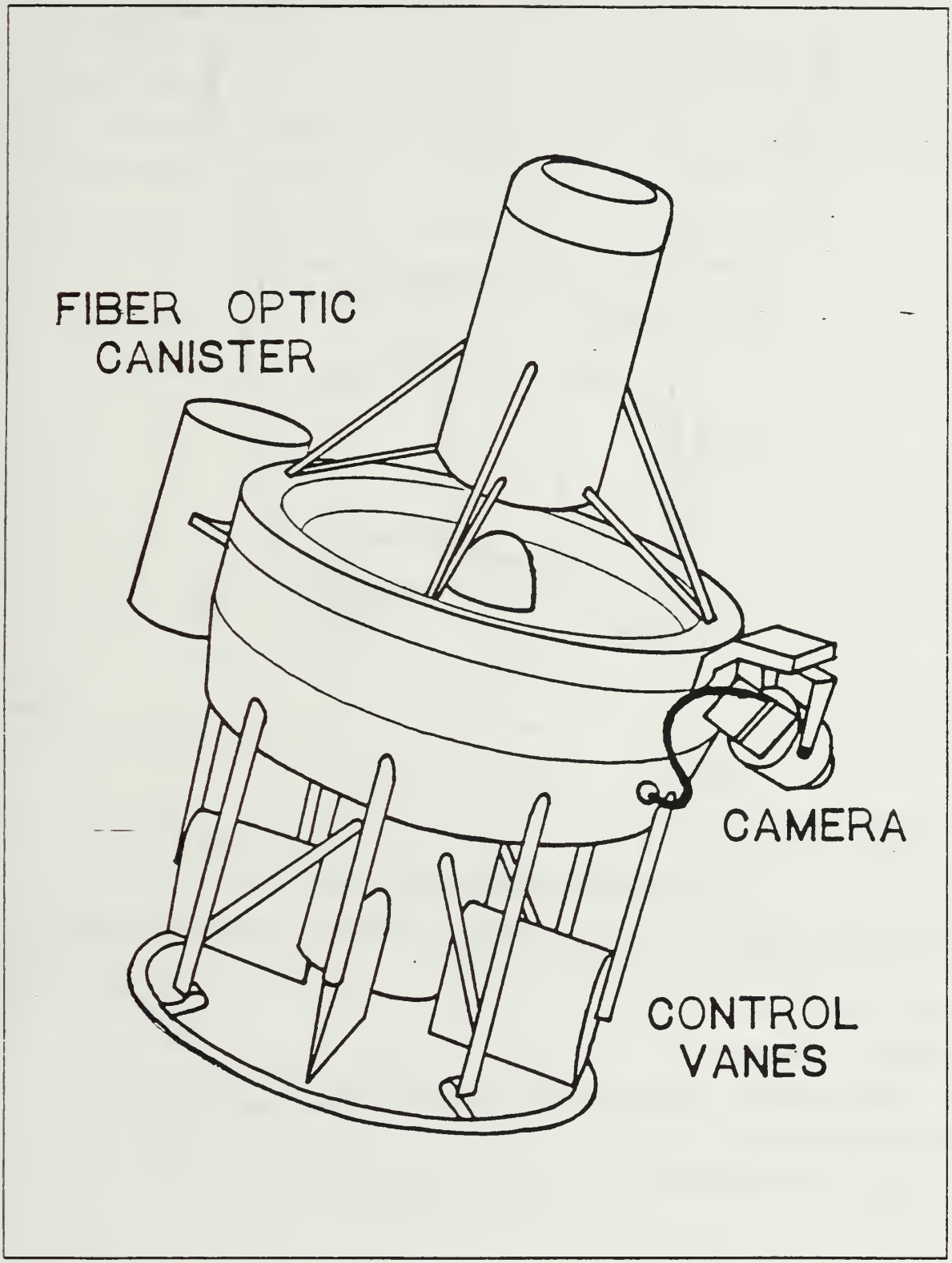

Figure 1.1 Schematic Drawing of AROD

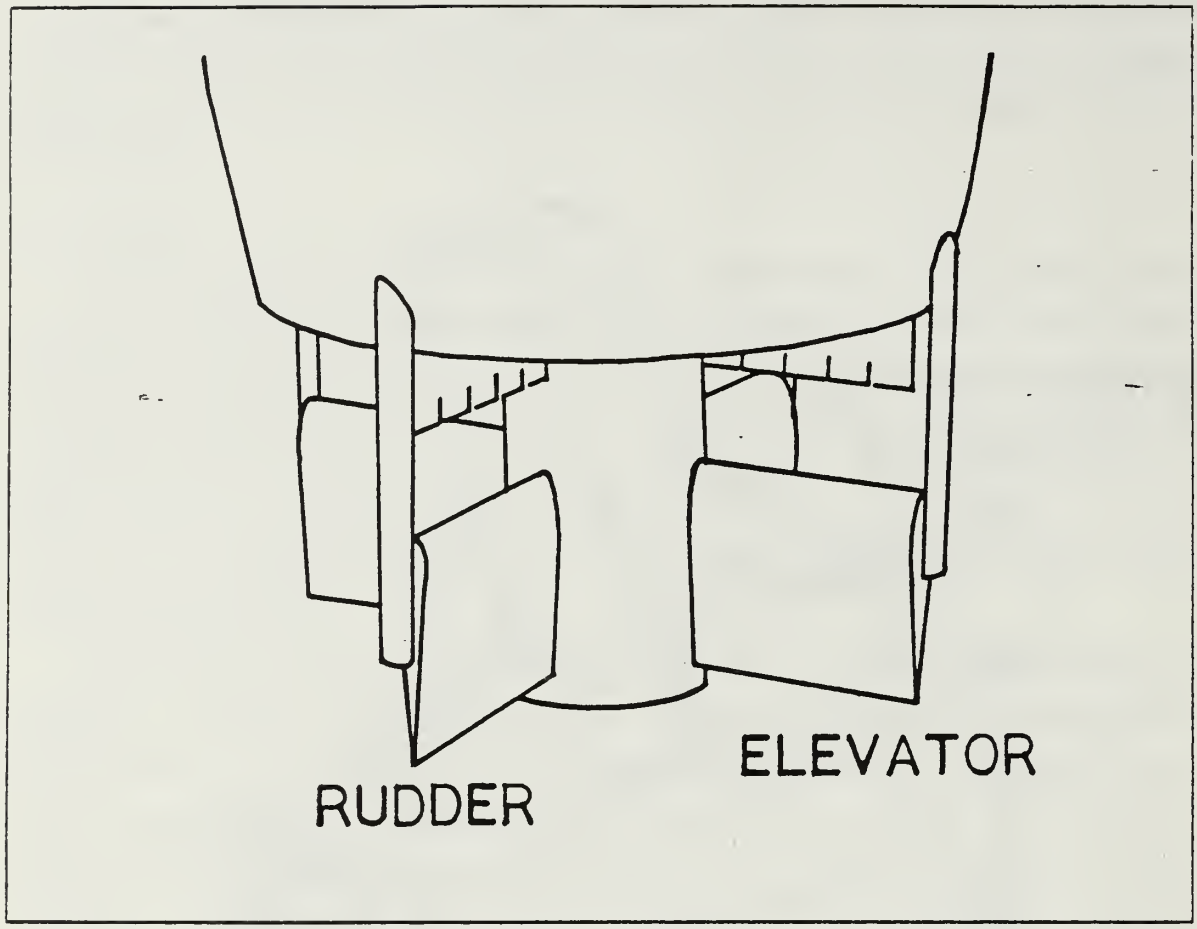

Figure 1.2 AROD Control Vanes

Vertical flight control of AROD is accomplished through a throttle controller which incorporates the same type of model airplane servo used to actuate the aerodynamic control vanes. The throttle controller increases or decreases the engine RPM as required to raise or lower AROD vertically.

C. THE PROBLEM

AROD is interesting from the standpoint of a control system design for several reasons. Most significant is the phenomenon of cross-coupled dynamics between the pitch and yaw subsystems. In addition, AROD is a Multi-Input Multi-Output system. These topics are briefly discussed in the following sections.

1. **Gyroscopic Coupling**

AROD's propeller has a spin velocity of 7200 RPM in the hovering condition. This creates a large angular momentum vector along the spin axis of the propeller. Thus, AROD can be thought of as a large gyroscope with its angular momentum vector oriented perpendicular to the surface of the earth. Consider a cylindrical rotor

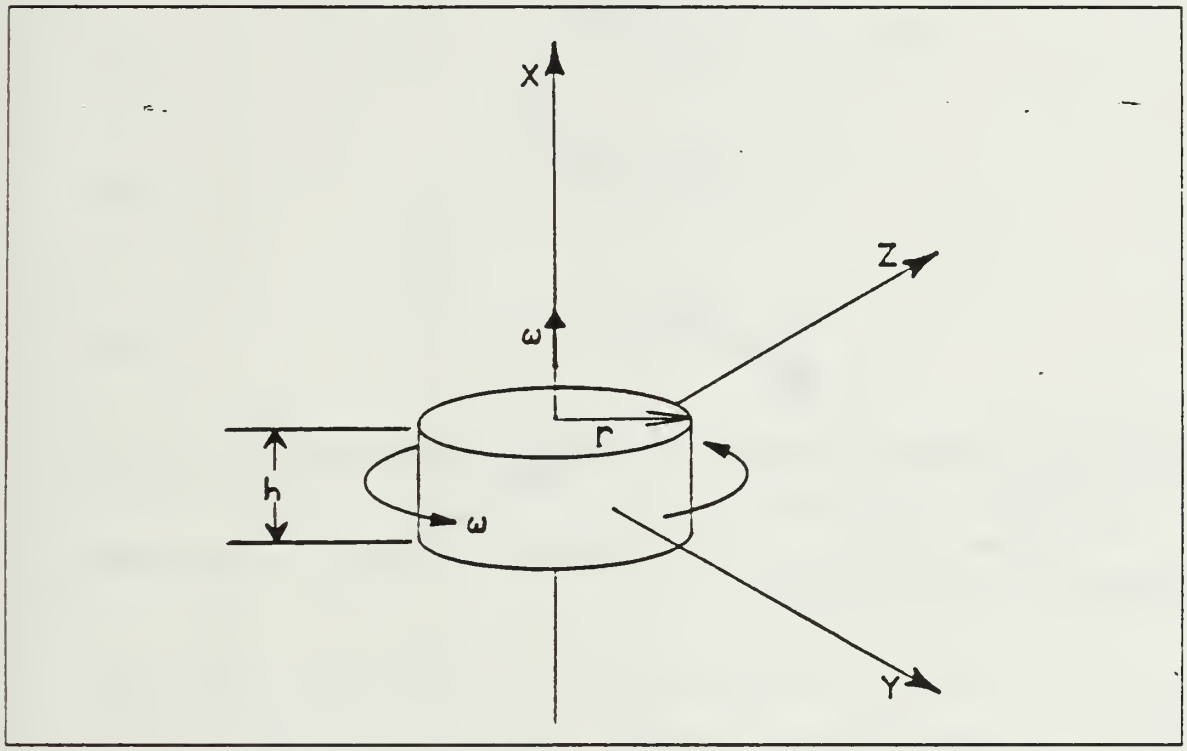

Figure 1.3 Spinning Rotor Orientation

spinning about the x-axis with an angular spin velocity ω. See Figure 1.3. Let the rotor be of mass, m, with moments of inertia I_x , I_y , and I_z about their respective axes. These moments are defined in the three equations below.

$$I_x = \iiint \delta(y^2 + z^2)\, dV \qquad (1.1)$$

$$I_y = \iiint \delta(x^2 + z^2)\, dV \qquad (1.2)$$

$$I_z = \iiint \delta(x^2 + y^2)\, dV \qquad (1.3)$$

In these equations, δ is the density of the rotor and dV is an incremental volume element of the rotor. In the case of AROD, it is assumed that the propeller is spinning with sufficient angular velocity that its dynamics can be approximated by the dynamics of a cylindrical disk having the same mass, m, radius, r, and thickness, h. The moments of inertia of the propeller then reduce to the following :

$$I_x = \frac{mr^2}{2} \tag{1.4}$$

$$I_y = \frac{m(3r^2 + h^2)}{12} \tag{1.5}$$

$$I_z = \frac{m(3r^2 + h^2)}{12} \tag{1.6}$$

Notice that the moments of inertia about the y-axis and z-axis are equal to each other due to the symmetry of the problem.

The angular velocity, ω , of the rotor induces an angular momentum vector, $\mathbf{H}_x$, defined by the following equation.

$$\mathbf{H}_x = I_x \omega \tag{1.7}$$

If a coupled pair of forces, **F**, directed parallel to the z-axis is applied to the the spin axis of the rotor at a distance, d, from the rotor's center of gravity, as in Figure 1.4, then a torque, **M** , results. The torque is given by

$$\mathbf{M} = \mathbf{F} \times \mathbf{d} \tag{1.8}$$

If the rotor were not spinning with angular velocity, ω , then this applied torque would result in rotation of the rotor about the y-axis. The angular momentum of the spinning rotor, however, results in quite a different response to the applied torque. According to Newton's Second Law, an external force applied to the center of gravity of a rigid body results in a change in the velocity of that body. A corresponding change in the body's momentum also results. The changes in velocity and momentum

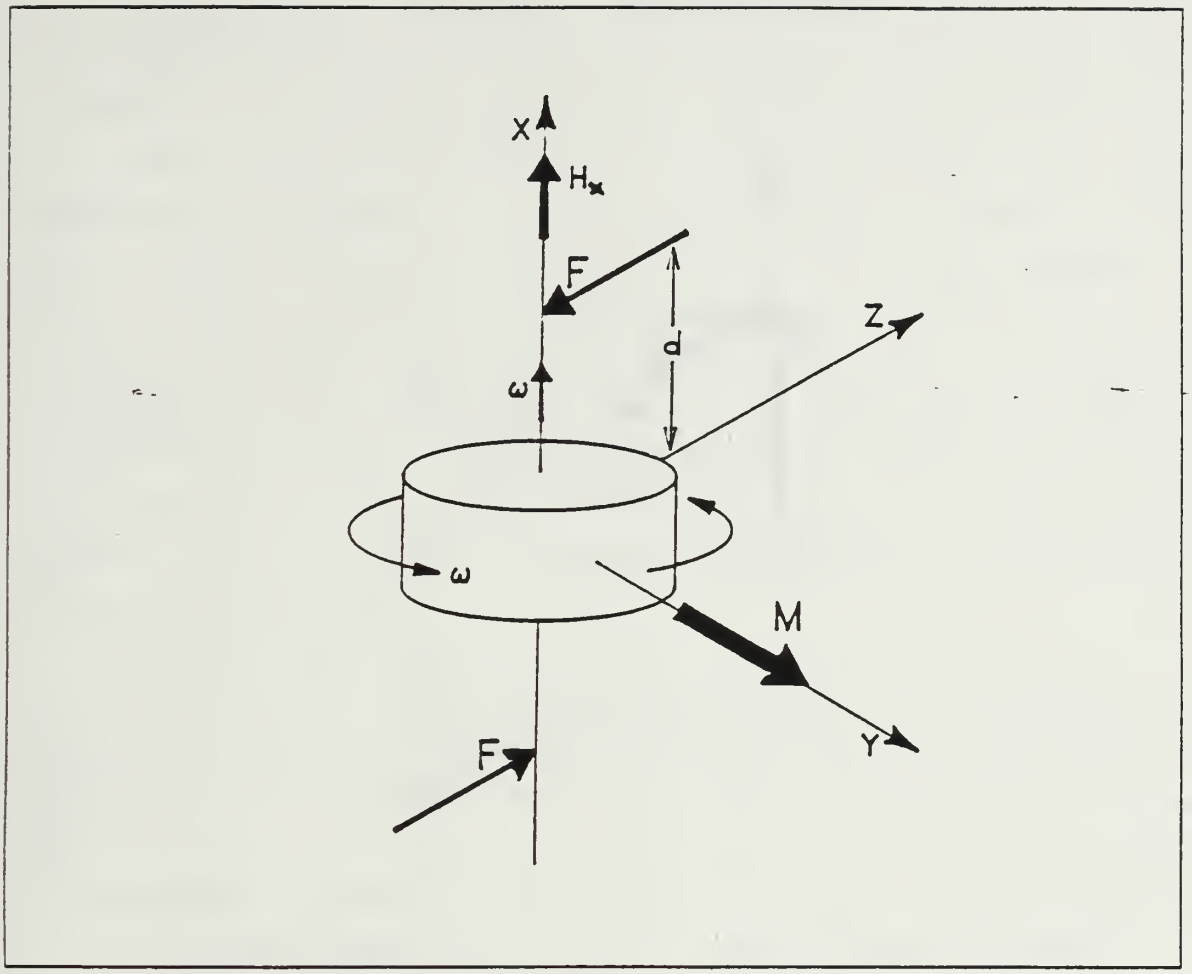

Figure 1.4 Spinning Rotor with a Force Couple Applied

are in the direction of the applied force. This law extends into the realm of angular forces, or moments, and angular velocities. In short, an applied moment, **M** , results in a change in angular momentum.

$$\mathbf{M} = \frac{d\mathbf{H}_x}{dt} = \frac{dI_x\,\omega}{dt} \qquad (1.9)$$

Notice that the change in angular momentum is in the same direction as the applied moment. This is the key to gyroscopic precession. By vector addition of $\mathbf{H}_x$ and **M** , it can be seen that a new angular momentum vextor, $\mathbf{H}_{new}$, results. The new angular momentum is displaced by an angle, ψ, from the initial angular momentum vector, $\mathbf{H}_x$. This angular movement is called precession and it occurs at a precession rate, **r**, oriented as shown in Figure 1.5 and defined by Equation 1.10.

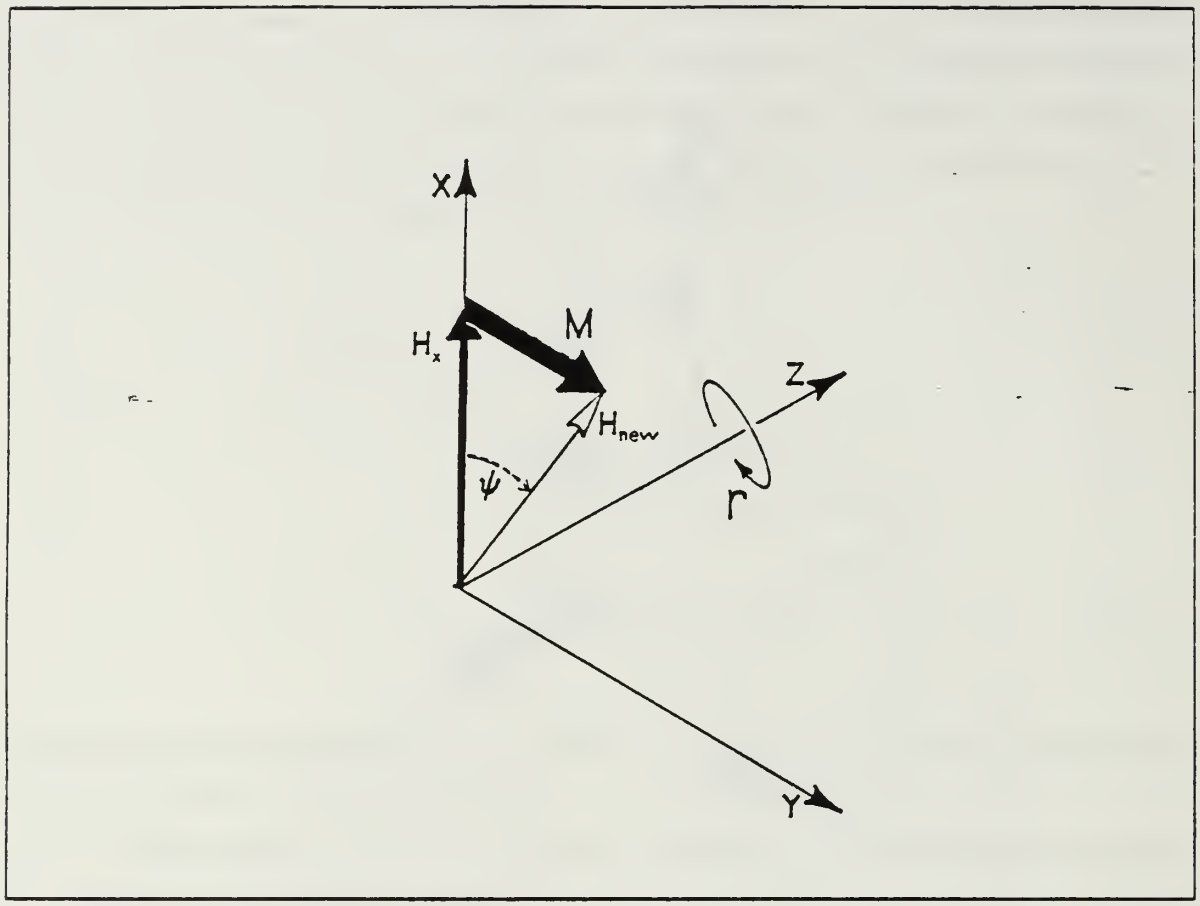

Figure 1.5 Resultant Angular Momentum With an Applied Torque

$$\mathbf{r} = \frac{d\psi}{dt} \tag{1.10}$$

It can be shown that $\mathbf{H}_x$, $\mathbf{M}$, and $\mathbf{r}$ are always mutually perpendicular to each other [Ref. 2: p. 335], and that these vectors are related by the expression

$$\mathbf{M} = I_x \mathbf{r} \times \mathbf{H}_x \tag{1.11}$$

The handy mnemonic for this relationship is that "spin follows torque". In this example, the spin vector, $\mathbf{H}_{new}$, that results from the applied torque, $\mathbf{M}$, is rotated in the direction of the torque. Notice that the magnitude of the precession velocity is directly proportional to the magnitude of the applied torque.

In the case of designing a control system for AROD, the preceding development is quite important. Recall that the propeller spins at a nominal velocity of 7200 RPM. The angular momentum of this high speed rotor has significant effect on the flight dynamics of AROD. In order to change the orientation of this angular momentum, as is required to accomplish translational flight, a considerable torque must be applied. This torque is produced by the four control vanes located in the propeller downwash. Note that the gyroscopic nature of AROD introduces cross-coupling of the pitch and yaw dynamics. For instance, when a pitching torque is commanded about the y-axis via the elevator vanes, AROD must first undergo an initial yawing motion about the z-axis. Similarly, a yaw command from the rudder vanes results in an initial pitching motion about the y-axis. These cross-coupled dynamics must be considered by the engineer when designing the controller for the elevator and rudder vanes.

2. **Multiple Control Loops**

Another feature which makes AROD interesting for the control engineer is the Multi-Input Multi-Output (MIMO) nature of its dynamics. There are basically four subsystems which need to be controlled in order to make AROD fly. These subsystems are :

1. Roll rate
2. Rate of vertical climb
3. Pitch angle
4. Yaw angle

Classical design tools such as Root Locus diagrams and Bode plots are not well suited for MIMO system design. Instead, the usefulness of these methods is primarily limited to Single-Input Single-Output (SISO) systems. These systems are generally represented in terms of their S domain or Z domain transfer functions. The poles and zeros of these transfer functions determine how the time response of the system will behave. By using the graphical and analytic methods available through classical design theory, the engineer can generally place the poles and zeros of his SISO controller in such locations as to obtain an acceptable time response for the system. The complex interactions that typically accompany a MIMO system can become impossible to represent in terms of standard transfer functions. Thus, classical design methods may become powerless for some systems. By developing a state space representation of the system, however, the interactions can be accurately modelled. Optimal control theory

is founded on the state space representation of control systems and, therefore, it seems logical to pursue this theory for AROD. The basics of optimal control theory are presented in the following chapter.

II. OPTIMAL CONTROL THEORY

A. FEEDBACK CONTROL

1. Why Use Feedback ?

Feedback control is familiar to engineers from all disciplines. In its simplist form, feedback control is nothing more than using the present condition, or "state", of a system to influence its condition in the future.

The advantages of state feedback control [Ref. 1: p.97] can be summarized in four points :

1. Assuming that controllability and observability conditions are satisfied, the transient time response of the system can be easily controlled and adjusted.
2. The sensitivity of the system to plant parameter variation is reduced.
3. Rejection of disturbance and noise signals is improved.
4. Steady state errors may be eliminated or reduced.

These benefits are not free. The penalty for using feedback control may include disadvantages such as :

1. System complexity increases because additional sensors may be required to measure the feedback states.
2. Sensors contribute to an increase in :
   a. Cost
   b. Size
   c. Weight
   d. Measurement noise
3. Closed loop gain is generally lower than open loop gain.

Despite these potential drawbacks, feedback systems are widely used in all engineering fields.

2. System Classification

Feedback provides a system with the ability to moniter and alter its performance. As the process advances in time, the system is apprised of the changes that occur in its states. This state information may be real time or may be delayed by

some finite interval of time. In general, systems may be separated into two categories according to the nature of the signals they process. These categories are :

1. Continuous time.
2. Discrete time.

An analog electrical circuit is one example of the first type of system. In this case, the voltage and current signals assume values over a continuum of time. That is, given any two instants in time, the changing values of these signals may be distinctly measured. This is true regardless of how closely the two time instants occur. Such systems are usually described by a series of differential equations. The Laplace transform is extremely useful in allowing frequency domain analysis and design of continuous time systems.

A microprocessor based system is an example of the discrete time system. The clocked signals in this type of system are represented by a sequence of numbers. Typically, a sequence of sampled data results from measuring an analog signal at specific intervals in time. The time between measurements is referred to as the sampling interval, or Δt. The sampling frequency, f_s, is simply the inverse of Δt. For an analog system with a Fourier transform bandlimited to a maximum frequency, f_{max}, the Nyquist frequency, f_n, is defined in Equation 2.1 [Ref. 3: p. 138].

$$f_n = 2f_{max} \tag{2.1}$$

A general rule of thumb for the design engineer [Ref. 4: p. 404] is to sample a system such that

$$f_s \geq 10f_n \tag{2.2}$$

This guideline for selecting a sampling frequency is based on the following considerations :

1. Most systems are not strictly bandlimited. The choice of a sampling frequency greater than the Nyquist frequency compensates for contamination by higher frequency disturbances [Ref. 5: p. 30].
2. The sampling frequency should be fast enough to avoid aliasing. This distortion is generated during the convolution reconstruction of a time signal from its Fourier transform [Ref. 3: pp. 135-137].

Discrete time systems are represented by difference equations and the Z-transform is the mechanism by which these equations are analyzed. More details of the comparisons between continuous time systems and discrete time systems will arise during subsequent discussion.

3. **System Structure with Feedback**

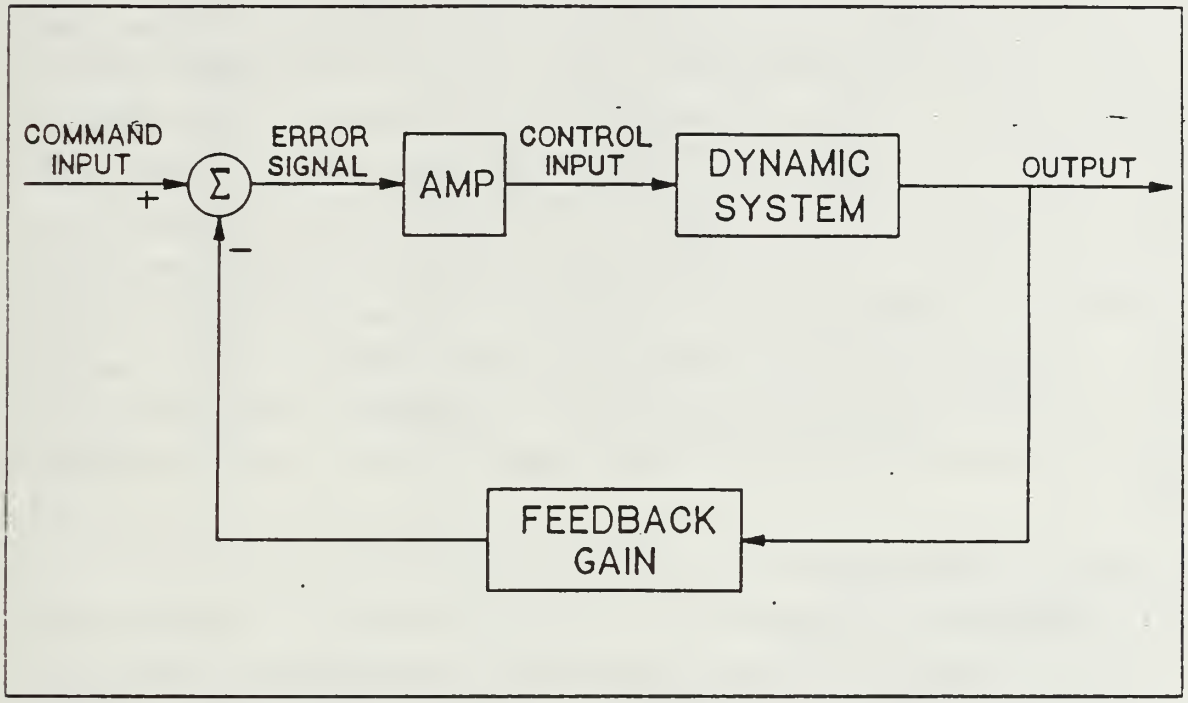

Figure 2.1 Basic Control System

The basic form for any control system is illustrated in Figure 2.1. It is the responsibility of the design engineer to determine any or all of the items designated in this schematic. In this section, emphasis is placed on the feedback gain "black box" shown in Figure 2.1. The two methods most commonly used in practice to determine feedback gains are pole placement techniques and optimal control techniques. The element of trial and error is inherent in both methods.

Pole placement techniques include analytical methods as well as the frequency domain methods previously mentioned. These methods are best suited to low-order, linear, time-invariant, SISO systems. Although these methods are extremely useful for certain problems, no detailed explanation of these classical techniques is included in this thesis.

Optimal control theory provides an alternative to classical pole placement techniques. A primary advantage of optimal control methods is that feedback gains can be computed for a much broader range of control problems. Specifically, optimal control provides solutions for high order, non-linear, time varying, MIMO systems. Such systems are intractable with classical methods. In addition, optimal control affords the designer the option to specify a performance criteria which is not linked to such standard time domain criteria as rise time, percent overshoot, and settling time. For instance, using optimal control theory, the design engineer may compute feedback gains which result in a system that responds in *minimum time* to a given command input. Selection of a different performance criteria might result in a system that responds with *minimum energy* expendiature, *minimum fuel*, or *minimum deviation* from the reference command. Sound engineering judgement and a thorough understanding of the system dynamics are prerequisites for effective application of optimal control theory. No guarantee is made that the feedback gains obtained by optimal control theory will result in acceptable system response. The designer should evaluate the system response and modify his performance criteria in order to achieve the desired output.

B. SYSTEM DEFINITION

1. Continuous Time Systems

The foundation of a successful control system is an accurate model of the plant which is being controlled. The state space representation of a general n^{th} order continuous time system is described by the following matrix state equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}(t)\,\mathbf{u}(t) \tag{2.3}$$

$$\mathbf{y}(t) = \mathbf{C}(t)\,\mathbf{x}(t) + \mathbf{D}(t)\,\mathbf{u}(t) \tag{2.4}$$

$$\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{r}(t) \tag{2.5}$$

$$\mathbf{u}(t) = \mathbf{F}(t)\,\{\mathbf{x}(t) - \mathbf{r}(t)\} \tag{2.6}$$

where the definitions in Table 1 apply to a system with ℓ control inputs and m measurable outputs.

TABLE 1

STATE SPACE DEFINITIONS FOR CONTINUOUS TIME SYSTEMS

Term	Dimension	Definition
$x(t)$	$(n \times 1)$	State vector
$u(t)$	$(\ell \times 1)$	Control input vector $(0 < \ell \leq n)$
$y(t)$	$(m \times 1)$	Output vector $(0 < m \leq n)$
$r(t)$	$(m \times 1)$	Command input vector
$e(t)$	$(m \times 1)$	Error vector
$A(t)$	$(n \times n)$	Plant matrix
$B(t)$	$(n \times \ell)$	Control distribution matrix
$C(t)$	$(m \times n)$	Output distribution matrix
$D(t)$	$(m \times \ell)$	Feedforward control gain matrix
$F(t)$	$(\ell \times m)$	State feedback gain yector

In this thesis, a linear time invariant system will be assumed. This allows the time dependency of the process matrices, $A(t)$ and $B(t)$, and the measurement matrices, $C(t)$ and $D(t)$, to be eliminated. Because optimal control theory requires that all n states be available for feedback, the output distribution matrix, C, is set equal to the identity matrix, I. This indicates that $m = n$ and that the state vector is completely observable. In addition, the feedforward control gain matrix, D, is assumed to be equal to the zero matrix. These assumptions lead to the following simplified state equations :

$$\dot{x}(t) = A\,x(t) + B\,u(t) \tag{2.7}$$

$$y(t) = x(t) \tag{2.8}$$

$$e(t) = x(t) - r(t) \tag{2.9}$$

$$u(t) = F(t)\,e(t) \tag{2.10}$$

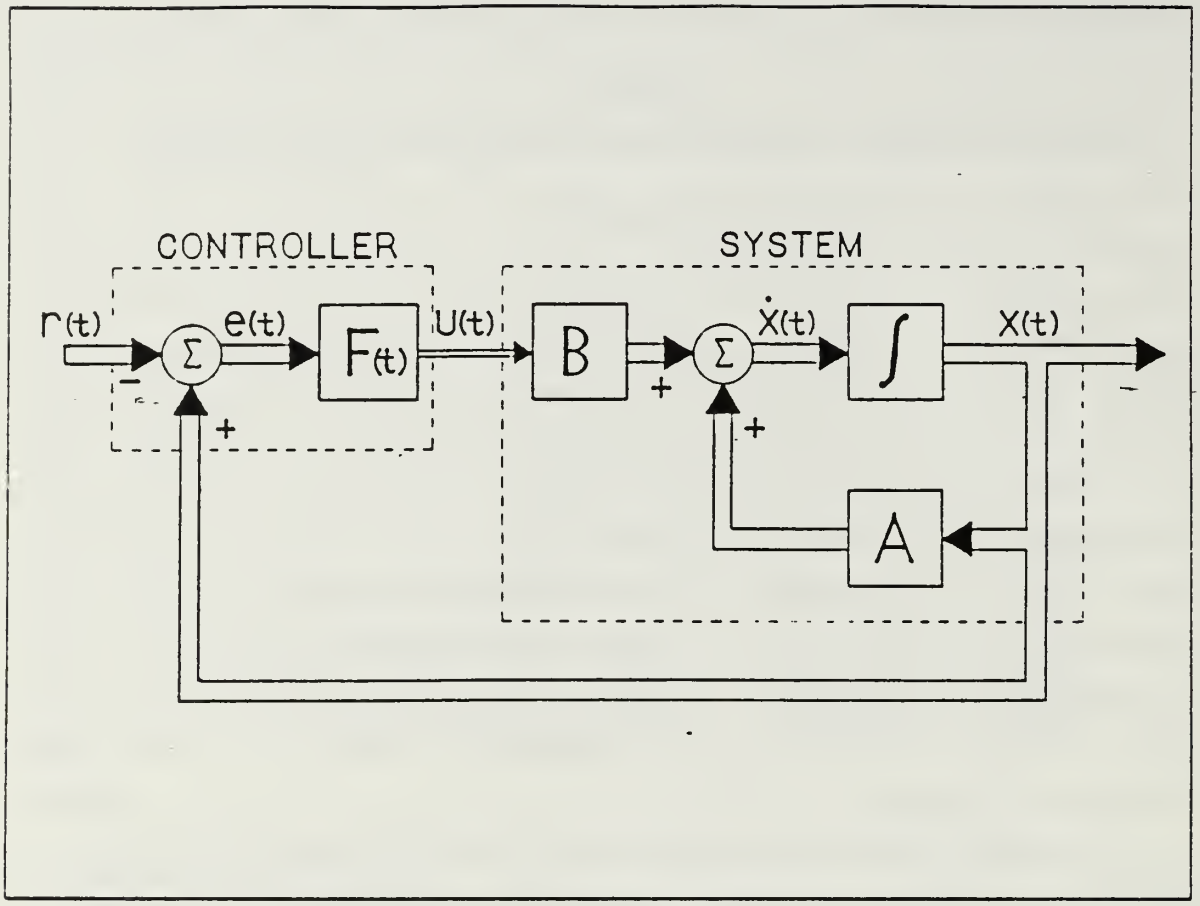

Figure 2.2 Continuous Time System

The realization of such a system is schematically illustrated in Figure 2.2.

2. **Discrete Time Systems**

Optimal control theory is applicable to the continuous time system presented in the preceeding section. The remainder of this thesis, however, will focus on the application of optimal control theory to sampled data systems. The motivation behind this effort is to develop an interactive, user-friendly software package that can be implemented on a microcomputer. The digital nature of sampled data systems make them ideal for analysis and design with these high speed computers. The theory for optimal control of discrete time systems is well developed and closely follows the development for continuous time systems [Ref. 6].

As was noted earlier, many digital systems are the result of periodic sampling of analog systems. This fact makes it necessary to mathematically connect the two types of systems. For an analog signal that is sampled at the frequency, f_s, a discrete

signal value is measured every $t = k\Delta t$ seconds. In this notation, k is an integer time index in the range $0 \leq k \leq (N-1)$ where N represents the last sample period of interest. Letting the sample period be denoted as

$$\Delta t = T \tag{2.11}$$

and substituting a discrete approximation for the derivative in Equation 2.7,

$$\dot{x}(kT) \sim \frac{x((k+1)T) - x(kT)}{T} \tag{2.12}$$

yields the discrete state equation :

$$x((k+1)T) \sim (I + AT)x(kT) + TBu(kT) \tag{2.13}$$

The analytic solution for the discrete problem is given by :

$$x(k+1) = \Phi\, x(k) + \Gamma\, u(k) \tag{2.14}$$

$$y(k) = x(k) \tag{2.15}$$

$$e(k) = x(k) - r(k) \tag{2.16}$$

$$u(k) = F(k)\, e(k) \tag{2.17}$$

where Φ and Γ are defined as :

$$\Phi = e^{AT} \tag{2.18}$$

$$\Gamma = \int_0^T e^{At}\, dt\, B \tag{2.19}$$

The vectors and matrices which describe an n^{th} order discrete time system with ℓ control inputs and m measured outputs are summarized in Table 2. A graphical realization of such a system is illustrated in Figure 2.3.

TABLE 2
STATE SPACE DEFINITIONS FOR DISCRETE TIME SYSTEMS

Term	Dimension	Definition
x(k)	(n × 1)	State vector
u(k)	(ℓ × 1)	Control input vector ($0 < \ell \leq n$)
y(k)	(m × 1)	Output vector ($0 < m \leq n$)
r(k)	(m × 1)	Command input vector
e(k)	(m × 1)	Error vector
Φ(k)	(n × n)	State transition matrix
Γ(k)	(n × ℓ)	Discrete Control distribution matrix
C(k)	(m × n)	Output distribution matrix
D(k)	(m × ℓ)	Feedforward control gain matrix
F(k)	(ℓ × m)	State feedback gain vector

The computation of the discrete process matrices, Φ and Γ, from the continuous process matrices, **A** and **B**, is readily accomplished [Ref. 5: p. 37] on a digital computer as follows.
Define an auxillary matrix, Π as

$$\Pi = \int_0^T e^{At}\, dt \qquad (2.20)$$

$$= IT + \frac{AT^2}{2!} + \frac{A^2T^3}{3!} + \cdots + \frac{A^iT^{(i+1)}}{(i+1)!} + \cdots$$

The terms in this Taylor series expansion are computed until the result is within a specified degree of accuracy. It behooves the programmer to set a very small tolerance

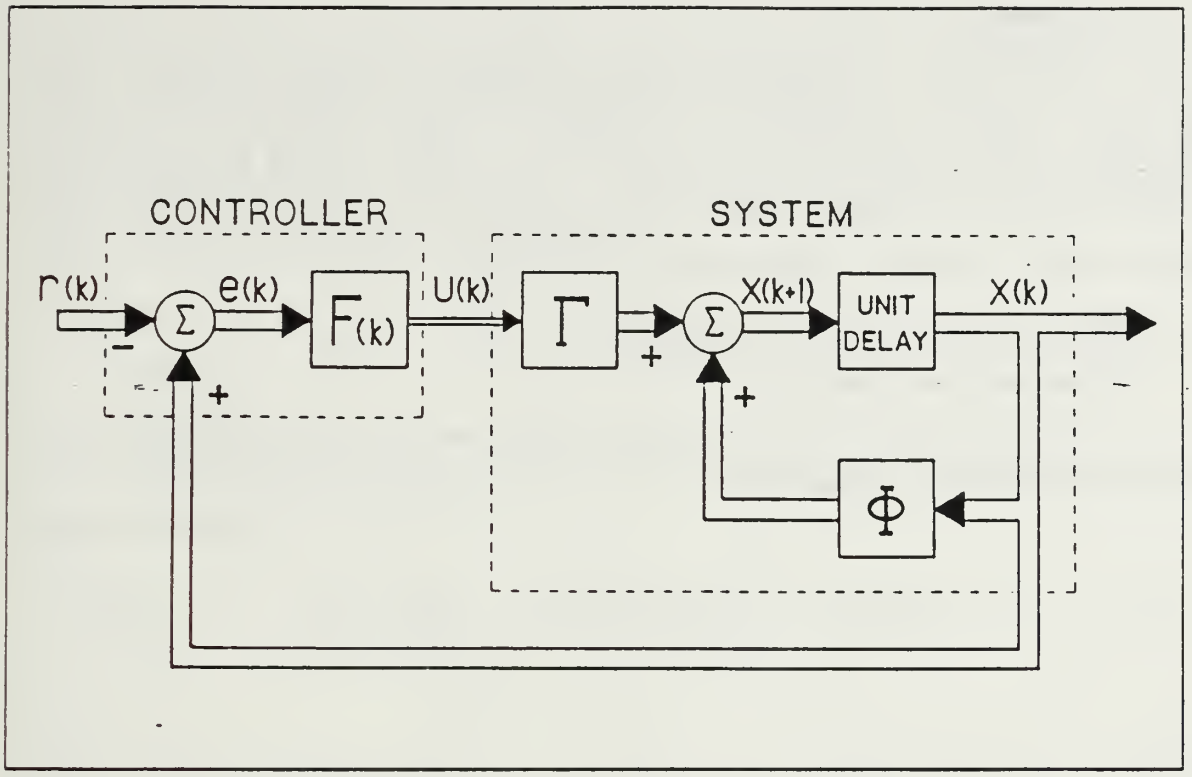

Figure 2.3 Time Invariant Discrete Time System

on the difference between successive terms in the expansion since this calculation is the critical link between the **A** and **B** matrices of the continuous time system and the Φ and Γ matrices of the discrete time system. Note that this calculation need only be done once for a given system with a fixed sample interval, T. The link is completed by using Equations 2.21 and 2.22.

$$\Phi = I + A\Pi \tag{2.21}$$

$$\Gamma = \Pi B \tag{2.22}$$

The subroutine PHIDEL listed in Appendix C performs the calculations required to solve Equations 2.20, 2.21, and 2.22. A tolerance of 10^{-7} is used for the allowable error between the last term used from the Taylor expansion and the first term not used.

3. Constraints

The system is now defined in terms of the Φ and Γ state space matrices. The next step in the design process is to identify any constraints under which the system will operate. These constraints are as unique to the problem at hand as is the system itself. Therefore, no detailed discussion on constraints is appropriate without first defining a specific problem. This is done in Chapter III.

C. THE PERFORMANCE MEASURE

1. Quadratic Cost Function

The central theme in discrete optimal control theory is minimization of a cost function, J, defined in Equation 2.23.

$$J = e^t(N)He(N) + \sum_{k=0}^{N-1}\left[e^t(k)Qe(k) + u^t(k)Ru(k)\right] \qquad (2.23)$$

where

J	=	Scaler cost of operating the system over the time interval $0 \leq k \leq (N-1)$
e(N)	=	State vector at the terminal time
e(k)	=	State vector at intermediate discrete times
u(k)	=	Control vector at intermediate discrete times
H	=	Terminal state weighting matrix
Q	=	State trajectory weighting matrix
R	=	Scaler control weighting matrix
N	=	Time index at terminal time
t	=	Matrix transpose operator

2. Regulators and Trackers

It is imperative to note here that the *error* state vector, **e**(k), in Equation 2.23 *may not* be the same as the *system* state vector, **x**(k), that is presented in the previous section. The **e** state vector is usually formulated in one of two ways :

1. If **e**(k) = **x**(k), then the problem is a 'regulator' problem. The objective is to drive the system states to the origin during the time interval (1,N).
2. If **e**(k) = **x**(k) - **r**(k), then the problem is a 'tracking' problem. The objective is to drive the system states to have minimum deviation from the command input, **r**(k), during the time interval (1,N).

In order to extend the regulator problem to the tracking problem, the command input vector, $\mathbf{r}(k)$, must contain the same eigenvalues and structure as the $\mathbf{x}(k)$ state vector. It is possible, in many problems, to structure an approximate $\mathbf{r}(k)$ vector so that the use of the regulator solution is allowed. In the case of the regulator, the control input signal is generated as shown in Equation 2.24.

$$\mathbf{u}(k) = \mathbf{F}(k)\,\mathbf{x}(k) \tag{2.24}$$

In the case of the tracking problem, the control signal is generated from the error signal as shown in Equation 2.25.

$$\mathbf{u}(k) = \mathbf{F}(k)\,\{\mathbf{x}(k) - \mathbf{r}(k)\} \tag{2.25}$$

The comparison between these two types of systems is demonstrated in their block diagram structures as shown in Figure 2.4.

3. **Performance Weighting Factors**

The **H**, **Q**, and **R** weighting matrices are the parameters by which the design engineer shapes the solution of an optimal control problem to best suit the problem. There are no magic formulas which instruct the designer on how to choose the values of these parameters. Intuition, experience, and patience are the keys to successful design. It is in selecting values for these performance weighting factors that the process of trial and error enters the design process. There are, however, some restrictions and general guidelines that partially direct the efforts of a design engineer.

First consider the restrictions. Both of the state weighting matrices, **H** and **Q**, must satisfy all of the criteria below [Ref. 7: p. 753].

1. Dimension is $(n \times n)$
2. Real
3. Symmetric
4. Positive semidefinite

In addition, the designer should *never* allow both **H** and **Q** to be equal to the zero matrix at the same time. The resulting cost function would be

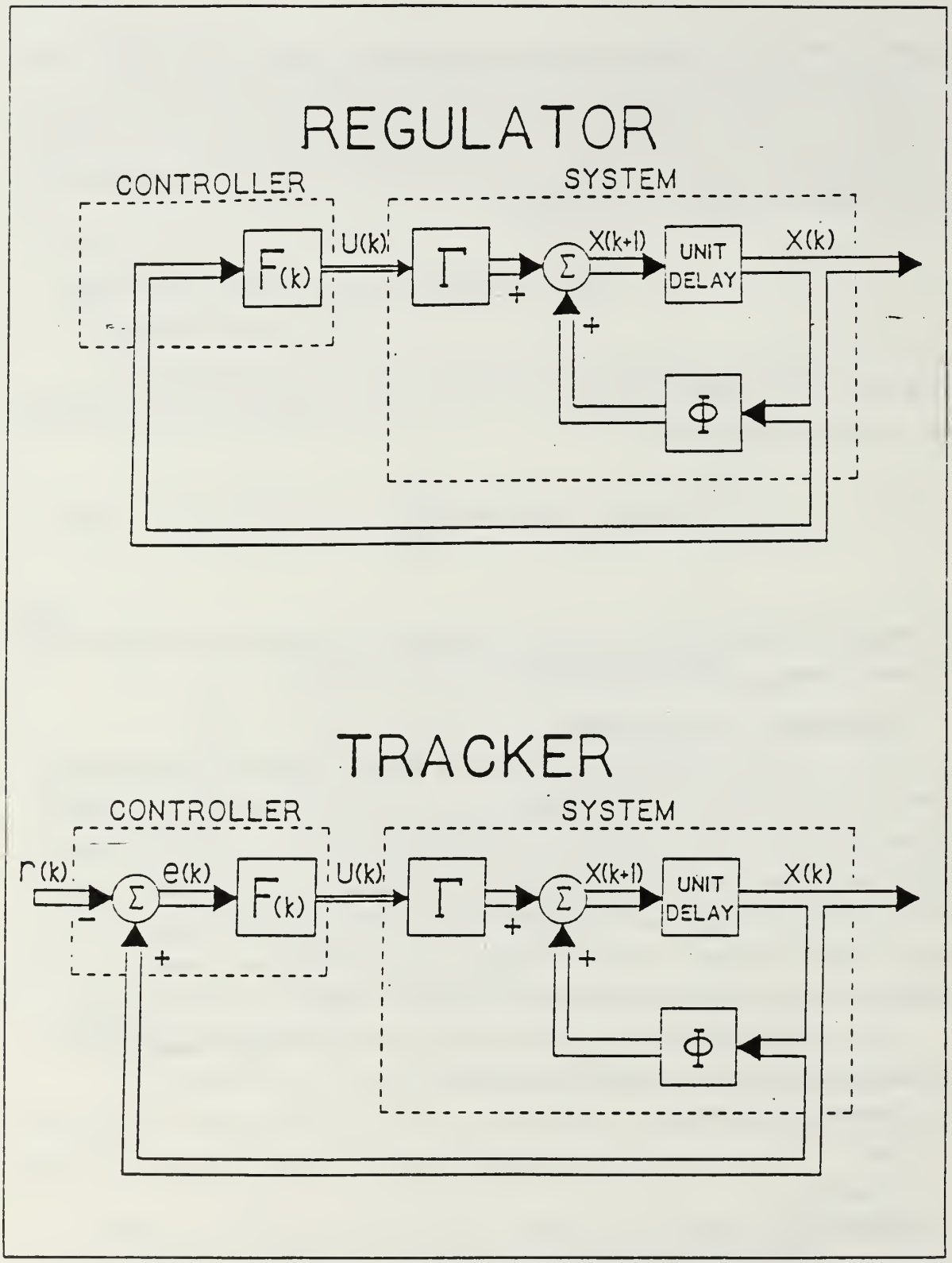

Figure 2.4 Comparison Between Regulator and Tracker System Structure

$$J = \sum_{k=0}^{N-1} \mathbf{u}^t(k)\mathbf{R}\mathbf{u}(k) \qquad (2.26)$$

It is simple to see that the cost function will be minimized by setting $\mathbf{u}(k)$ equal to zero [Ref. 7: p 755]. This would be disasterous since there would be no command signal available to drive the system states toward the desired state. It is permissible to set *either* **H** or **Q** equal to the zero matrix provided that they are *never both zero* simultaneously. The ($\ell \times \ell$) control weighting matrix, **R**, must be positive definite in order to assure the existence of a finite control [Ref. 7: p. 754].

Although there is no requirement for **H** and **Q** to be diagonal matrices, the usual practice is to select non-negative values for their diagonal elements and to set all off-diagonal elements to zero. This technique eliminates all cross product terms in the cost function. For example, consider a second order system containing states x_1 and x_2. If diagonal matrices are selected for **H** and **Q**, then only the $(x_1)^2$ and $(x_2)^2$ terms will be weighted in the cost function. There will be no consideration given to the x_1x_2 or x_2x_1 cross product terms.

Assuming that neither **H** nor **Q** is the zero matrix, it is suggested that the elements of these weighting matrices be selected so as to normalize the values of the states which they multiply [Ref. 6: p. 32]. For instance, suppose that x_1 represents the RPM of AROD's propeller and x_2 represents the angular displacement in degrees of the throttle servo. State x_1 is expected to have a nominal value of 7200 while state x_2 may have a nominal value of only 10. Assume that element q_{11} which multiplies $(x_1)^2$ and element q_{22} which multiplies $(x_2)^2$ are both set equal to one in an attempt to weight the two states equally. In terms of the cost function, the RPM will be weighted much heavier (approximately 720^2 times heavier !!) than the throttle servo position angle. An appropriate solution is to set $q_{11} = (1/720)^2$ if $q_{22} = 1$. Thus, each signal is given equal consideration in the cost function.

Notice in Equation 2.23 that the terminal cost term is not included inside the $\sum$ operator. This means that the **H** term is only counted one time and therefore can contribute only once to the overall cost. The state trajectory term and the control term, however, are counted N times. If no compensation is made for this disparity, the cost of an error in the terminal state is likely to not have any impact on the control solution. It seems that an additional scaling is required on the weighting elements. If the summation in Equation 2.23 is to be made over 500 discrete time intervals, for

instance, the normalized elements of **H** should be multiplied by 500 in order to weight the terminal states on the same scale as the state trajectory and control effort terms. Alternately, the elements of **Q** and **R** could be divided by 500. It is the *relative* magnitudes, not the *absolute* magnitudes, of these weighting factors which shape the control solution.

4. **Specific Types of Problems**

Optimal control theory can be used to solve any of the following types of problems :

1. Minimal time
2. Minimal control effort
   a. Minimum fuel
   b. Minimum energy
3. Minimal error from a reference
   a. Regulator
   b. Tracking
   c. Terminal state control

Each of these problem types requires minimization of a unique cost function in order to generate the appropriate control solution [Ref. 6: pp. 30-34]. The cost functions which are to be explored in this thesis are listed in Table 3.

TABLE 3
TYPICAL COST FUNCTIONS

Goal	Cost Function	Additional Explanation
Regulator	$J_1 = \sum \mathbf{x}^t(k)\mathbf{Q}\mathbf{x}(k)$	
Tracker	$J_2 = \sum \mathbf{e}^t(k)\mathbf{Q}\mathbf{e}(k)$	$\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{r}(k)$
Terminal Control	$J_3 = \mathbf{e}^t(N)\mathbf{H}\mathbf{e}(N)$	$\mathbf{e}(N) = \mathbf{x}(N) - \mathbf{r}(N)$
Minimum Energy	$J_4 = \sum \mathbf{u}^t(k)\mathbf{R}\mathbf{u}(k) + J_3$	
Minimum Fuel	$J_5 = \sum \lvert \mathbf{u}(k) \rvert$	

All $\sum$ are performed over the interval $0 \leq k \leq (N-1)$.

Of course, the cost function in Equation 2.23 is a general form which embodies all of the specific cost functions (except for minimum fuel) contained in Table 3. Proper selection of **H**, **Q**, and **R** will allow calculation of feedback control gains which cause the system to meet the specified performance goal.

D. CALCULATION OF OPTIMAL FEEDBACK GAINS

The method of dynamic programming is the workhorse which permits the calculation of optimal feedback control gains. Developed in the late 1950's by R.E. Bellman, this design tool provides a closed form solution for the minimization of the quadratic cost function for a discrete time linear system [Ref. 6: p. 84].

The procedure involved in calculating optimal control gains is unique in that the computation begins with the final, or N^{th}, discrete time interval and progresses *backwards in time* to the previous interval of the system process. This procedure in 'negative time' is possible because the calculation of the gains do not require any information about the state vector, **x**(k). The sequence of calculations continues in negative time until a separate gain matrix is determined for every discrete sample period in the time interval (0,N). The resulting time-dependent gain trajectory is usually stored in memory so that the control signal, **u**(k), may be computed.

An interesting and very useful property of the gain trajectory is its tendency to approach a constant valued matrix, $\mathbf{F}_{ss}$, under certain conditions [Ref. 4: p. 354]. This constant matrix is referred to as the *steady state* feedback gain matrix. The conditions necessary for $\mathbf{F}_{ss}$ to exist are :

1. The system is controllable.
2. The $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ Matrices are time invariant.
3. The **H** terminal state weighting matrix is equal to the zero matrix.
4. The **Q** trajectory state weighting matrix is constant.
5. The **R** control weighting matrix is constant.
6. The number of stages, N, of the process is *large.*

It is possible for the feedback gain matrix to approach $\mathbf{F}_{ss}$ without satisfying the first three conditions above. When all conditions are satisfied, however, a steady state gain solution is guaranteed provided that N is large enough. Just how large N must be in order to allow the gain trajectory to reach steady state is determined by the slowest time constant in the solution of the discrete matrix Riccati equation [Ref. 5: p. 259]. In practice, trial and error is the the most expedient method to determine how large N must be in order to ensure a steady state gain matrix, $\mathbf{F}_{ss}$.

A series of three recursive equations [Ref. 6: p. 83] is used to calculate the gain trajectory, F(k). It is convenient to introduce a *negative time* discrete index, K, which is defined as follows

$$K = N - k \tag{2.27}$$

Since k varies from (0,N-1) as forward time progresses, K varies from (1,N) as negative time progresses. Equation 2.28 is the solution for the transpose of the optimal feedback gain vector at each discrete time step. This equation is the solution to the well known discrete matrix Riccati equation. Equations 2.29 and 2.30 are auxillary equations required to complete the calculations. The recursive matrix equations are :

$$F(K) = -\{\Gamma^t P(K-1)\,\Gamma + R\}^{-1}\{\Gamma^t P(K-1)\,\Phi\} \tag{2.28}$$

$$\Psi(K) = \Phi + \Gamma F(K) \tag{2.29}$$

$$P(K) = \Psi^t(K) P(K-1)\,\Psi(K) + Q + F^t(K)R\,F(K) \tag{2.30}$$

with 'negative time' initial conditions

$$F^t(0) = 0 \tag{2.31}$$

$$\Psi(0) = 0 \tag{2.32}$$

$$P(0) = H \tag{2.33}$$

While somewhat difficult at first glance, Equations 2.28, 2.29, and 2.30 are ideal for iterative solution by a digital computer. These equations are solved in the main OPTCON program listed in Appendix B. The reader who is not familiar with OPTCON is encouraged to review the discussion of this program in Appendix A prior to continuing with Chapter III.

III. CONTROL SYSTEM DESIGN FOR AROD

A. OVERVIEW

The purpose of this chapter is to use optimal control theory to design an automatic flight control system for the U.S. Marine Corps' remotely piloted AROD. In their initial form, the equations of motion which describe AROD's dynamic behavior are extremely nonlinear and present a formidable challenge to the control system designer. For this reason, the equations are first linearized about a steady state hover condition. The restrictions and assumptions under which the linearized equations of motion are developed are summarized below:

1. The mass of AROD is constant with time [Ref. 8: p. 11].
2. The propeller angular velocity is constant.
3. Perturbations from steady state hover are small. This restriction requires that AROD pitch, roll, and yaw angular displacements be limited to less than 15° ($\pi/12 = 0.2618$ radians) [Ref. 8: p. 41].
4. Steady state pitch and yaw rates are zero.
5. Initial side velocities are zero.
6. Initial bank angles are zero.
7. Initial angular velocities are zero [Ref. 8: p. 45].

In order to elucidate the equations of motion, Figure 3.1 is provided for reference. The axis system in Figure 3.1 is known as a body-fixed coordinate system. The body-fixed axis system is thought of as being rigidly attached to the vehicle so that any change in the orientation of AROD with respect to the earth-fixed axis system (X′, Y′, Z′) results in a corresponding change in the orientation of the body-fixed axis system (X, Y, Z) with respect to (X′, Y′, Z′). The angles φ, θ, and ψ, commonly known as the Euler angles [Ref. 8: p. 25], are measures of the roll, pitch, and yaw angles respectively between the body-fixed (X, Y, Z) coordinate system and the earth-fixed (X′, Y′, Z′) coordinate system. The angular velocities p, q, and r in Figure 3.1 correspond to the roll, pitch, and yaw rates respectively.

The automatic flight control system is logically separated into three subsystems according to the simplified equations of motion. The three control subsystems which

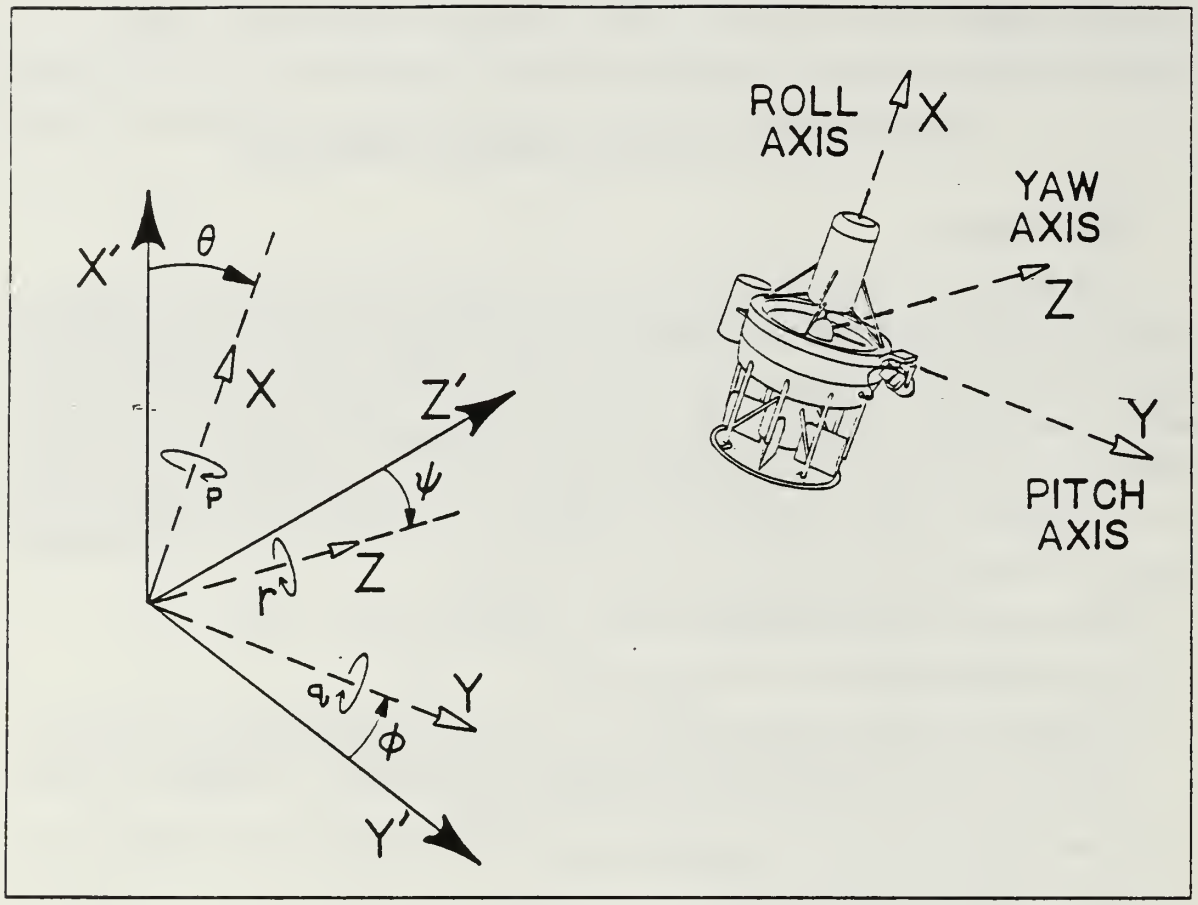

Figure 3.1 AROD Body-Fixed Coordinate System

are hereafter designed are :

1. Roll rate controller.
2. Altitude rate controller.
3. Pitch angle and yaw angle controller.

Each of these controllers is designed independently from the other subsystems. Therefore, any cross-coupling which may occur across the subsystem boundaries is not accounted for. Each of the following sections is devoted to the design of a controller for one specific AROD subsystem. A detailed design is first presented in Section B for the roll rate controller in order to demonstrate the iterative nature of the design process. Section C presents the results for the altitude rate controller. In the interest of brevity, only the initial trial run and the final solution for the altitude rate controller are presented. The coupled dynamics of AROD's pitch and yaw is examined in greater detail in Section D.

B. AROD ROLL RATE CONTROLLER

1. The Roll System

The purpose for roll rate control of the AROD is to allow the remote pilot to command a desired rotation velocity about the vehicle's longitudinal, or x, axis. Such movement allows the camera aboard the vehicle either to slowly scan a selected ground sector or to terminate the rolling motion so that a target of interest can be further studied. The nature of remote sensing requires that the vehicle respond rapidly to a roll rate command. When the remote pilot locates a ground target, he needs to be able to swiftly bring the vehicle to a zero roll rate condition with negligible overshoot. Such movement is commanded through a twistable handgrip control located on the pilot's console. It is assumed that this roll rate command, p_c, is limited to a step input of 1 radian/second (57.3°/second). Although no time response criteria are specified by the Marine Corps, it is assumed for the purpose of this work that the following design specifications for roll rate are required :

1. Zero steady state error for a step input is required.
2. The two percent settling time, $t_{2\%}$, is less than 1 second.
3. No overshoot is allowed.

The simplified equation of motion which describes AROD's roll rate subsystem is given in Equation 3.1.

$$\dot{p} = L_a \delta_a \tag{3.1}$$

The aileron servo dynamics are modelled in Equation 3.2 as a second order plant with a natural frequency, ω, of 2 Hz and a damping coefficient, ζ, of 0.707.

$$\ddot{\delta}_a = -2\zeta\omega\dot{\delta}_a - \omega^2\delta_a + \omega^2 u_a \tag{3.2}$$

The definitions in Table 4 apply to Equations 3.1 and 3.2. In order to apply the theory of optimal control, a suitable state space representation of the roll rate system must first be developed. Figure 3.2 presents the state space signal flow graph selected for this subsystem.

TABLE 4

VARIABLE DEFINITIONS FOR AROD ROLL RATE EQUATIONS OF MOTION

Variable	Definition	Value	Units
p	Vehicle Roll Rate	TBD	radians/second
L_a	Aileron Effectiveness Coefficient	-21.29	seconds^{-2}
δ_a	Aileron Servo Deflection Angle	$\leq 30°$	radians
$\dot{\delta}_a$	Aileron Servo Deflection Velocity	$\leq 50°$/sec	radians/second
ζ	Aileron Servo Damping Coefficient	0.707	unitless
ω	Aileron Servo Natural Frequency	12.57	radians/second
u_a	Control Input to Aileron Servo	TBD	volts

By designating the output of each integrator in Figure 3.2 to be a state, the following third order state equations are derived :

$$x = \begin{bmatrix} p \\ \delta_a \\ \dot{\delta}_a \end{bmatrix} \tag{3.3}$$

$$\dot{x} = \begin{bmatrix} 0 & -21.29 & 0 \\ 0 & 0 & 1 \\ 0 & -157.91 & -17.77 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 157.91 \end{bmatrix} u_a \tag{3.4}$$

$$u_a = F\{x - r\} \tag{3.5}$$

Assuming that a unit step roll rate is commanded, the command input vector becomes

$$r = \begin{bmatrix} p_c \\ \delta_{ac} \\ \dot{\delta}_{ac} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

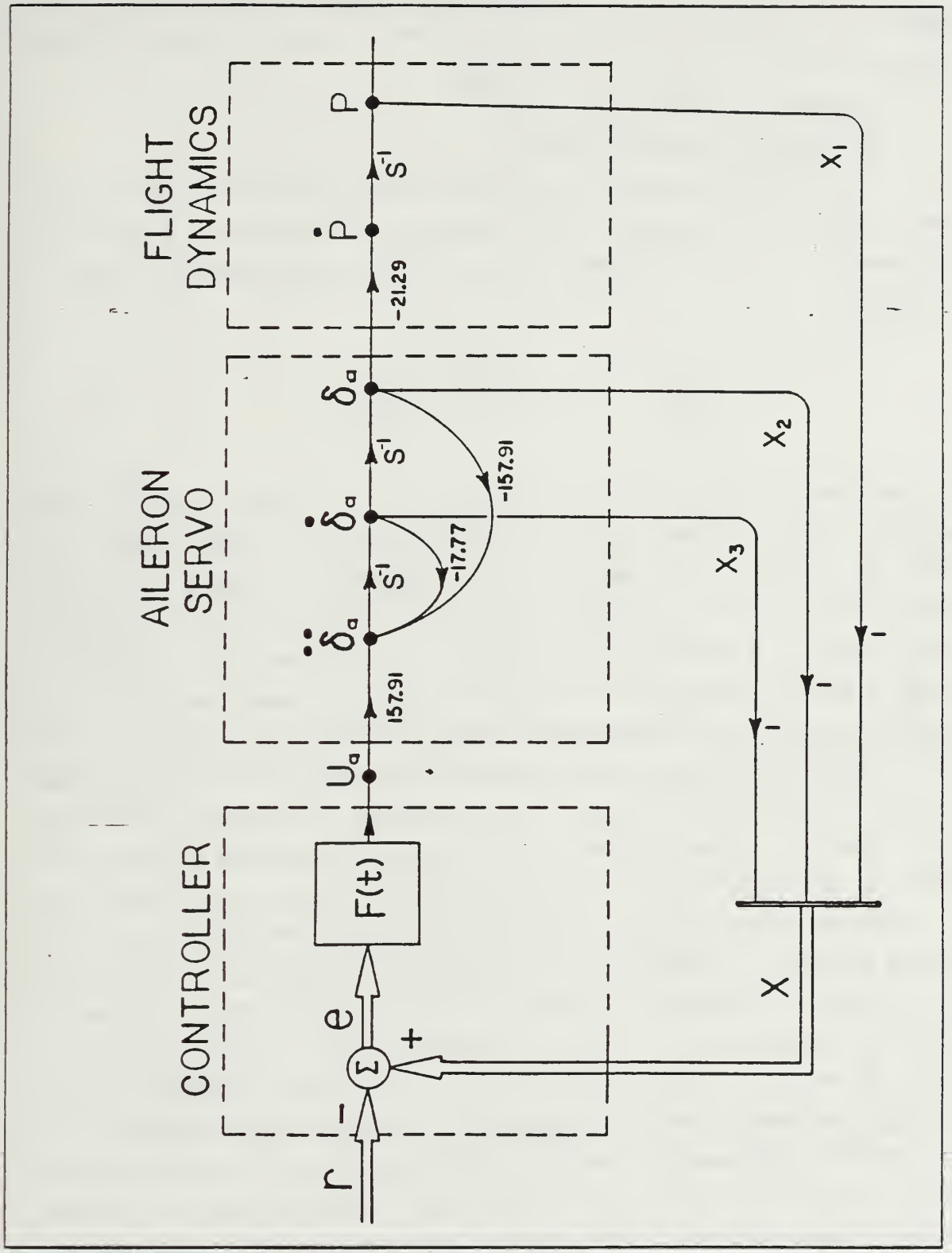

Figure 3.2 Signal Flow Diagram for Roll Rate Control

where the subscipt c indicates a command input variable. Now that the roll rate system and its constraints have been identified, the next step is to use the OPTCON program to design a suitable roll rate controller.

2. **Roll Rate Controller Design**

a. Choosing a Sampling Frequency

In order to determine an appropriate sampling rate for the roll rate system given in Equations 3.3 through 3.6, the bandwidth of the open loop system is first determined. The open loop transfer function for the roll rate system is given in Equation 3.7.

$$\frac{P(s)}{U_a(s)} = \frac{(157.91)\,(21.29)}{s^2 + 17.77\,s + 157.91} \qquad (3.7)$$

The open loop Bode diagram for this transfer function is shown in Figure 3.3. The negative 3 dB bandwidth of the magnitude curve is approximately 12 radians per second or 2 Hz. Using the criterion discussed in Chapter 2, The Nyquist sampling rate, f_n, is 4 Hz. In order to avoid aliasing effects and to ensure that the sampled system is a reasonable representation of the continuous time system, it is decided to employ a sampling frequency that is at least five times greater than f_n. A comparison of the effect of using various sampling frequencies is given in Table 5. The cost function which is used to obtain the optimal gains for this comparison is included in Table 5 and is hereafter referred to as the baseline cost function. The column labelled "Number of Stages Required" in Table 5 refers to the number of discrete time stages that must elapse before the optimal gain vector reaches its steady state value, $\mathbf{F}_{ss}$. Notice that the magnitude of the steady state gain vector is related to the sampling frequency. In general, a faster sampling rate yields larger feedback gains. Also notice that the sampling rate affects the amount of real time that is required for the gains to reach steady state. For example, when f_s is 20 Hz, it requires 1.60 seconds for $\mathbf{F}_{ss}$ to be achieved. When f_s is 40 Hz, however, a total time of 2.23 seconds elapses before $\mathbf{F}_{ss}$ is achieved. These considerations are important if the control gains are to be dynamically implemented. In the case of this design, the control implementation is limited to steady state values of the optimal gains. The unit step time response obtained by using steady state optimal feedback gains is observed to be nearly identical for all three of the sampling rates listed in Table 5. Specifically, the roll rate state, x_1,

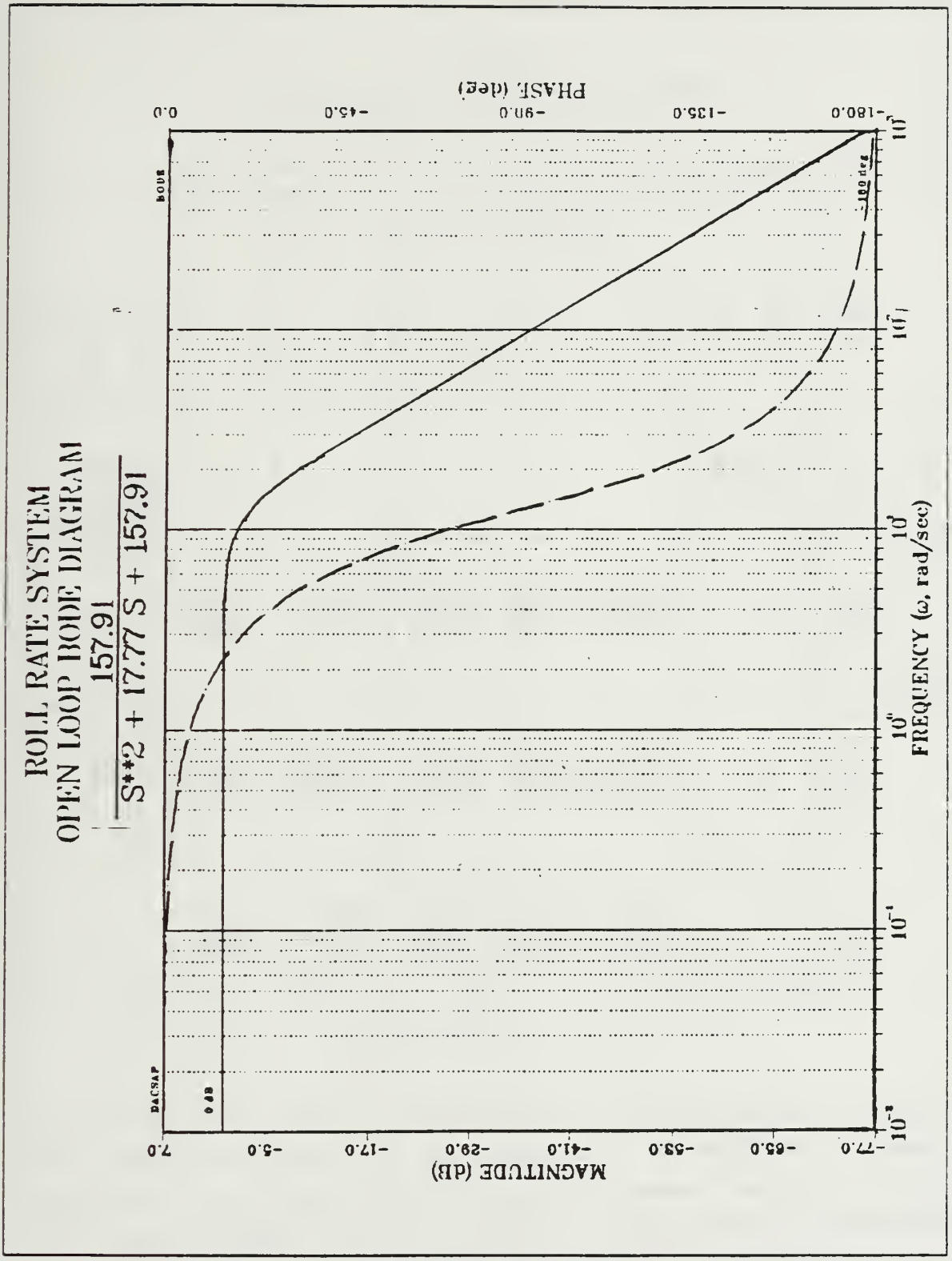

Figure 3.3 Open Loop Bode Diagram For Roll Rate System

TABLE 5

EFFECT OF SAMPLING FREQUENCY ON ROLL RATE SYSTEM

Baseline Cost Function

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = 1$$

Run	Sampling Frequency (Hz)	Φ	Γ	F_{ss}	Number of Stages Required
1	20	1.0000 -1.0086 -0.0196 0.0000 0.8547 0.0310 0.0000 -4.8985 0.3034	-0.0559 0.1453 4.8985	0.1696 -0.2866 -0.0866	32
2	40	1.0000 -0.5244 -0.0057 0.0000 0.9576 0.0199 0.0000 -3.1355 0.6047	-0.0078 0.0424 3.1355	0.2764 -0.9893 -0.1999	89
3	100	1.0000 -0.2124 -0.0010 0.0000 0.9926 0.0091 0.0000 -1.4430 0.8302	-0.0005 0.0074 1.4430	0.5126 -2.5612 -0.4400	159

time response exhibits an overshoot of approximately 3.7% and a 2% settling time of 1.3 seconds. See Figure 3.4. This implies that any of the three sampling rates examined is acceptable. Because it is generally considered good practice to implement small feedback gains when possible, it is decided to employ a sampling frequency of 20 Hz for the remainder of the roll rate controller design.

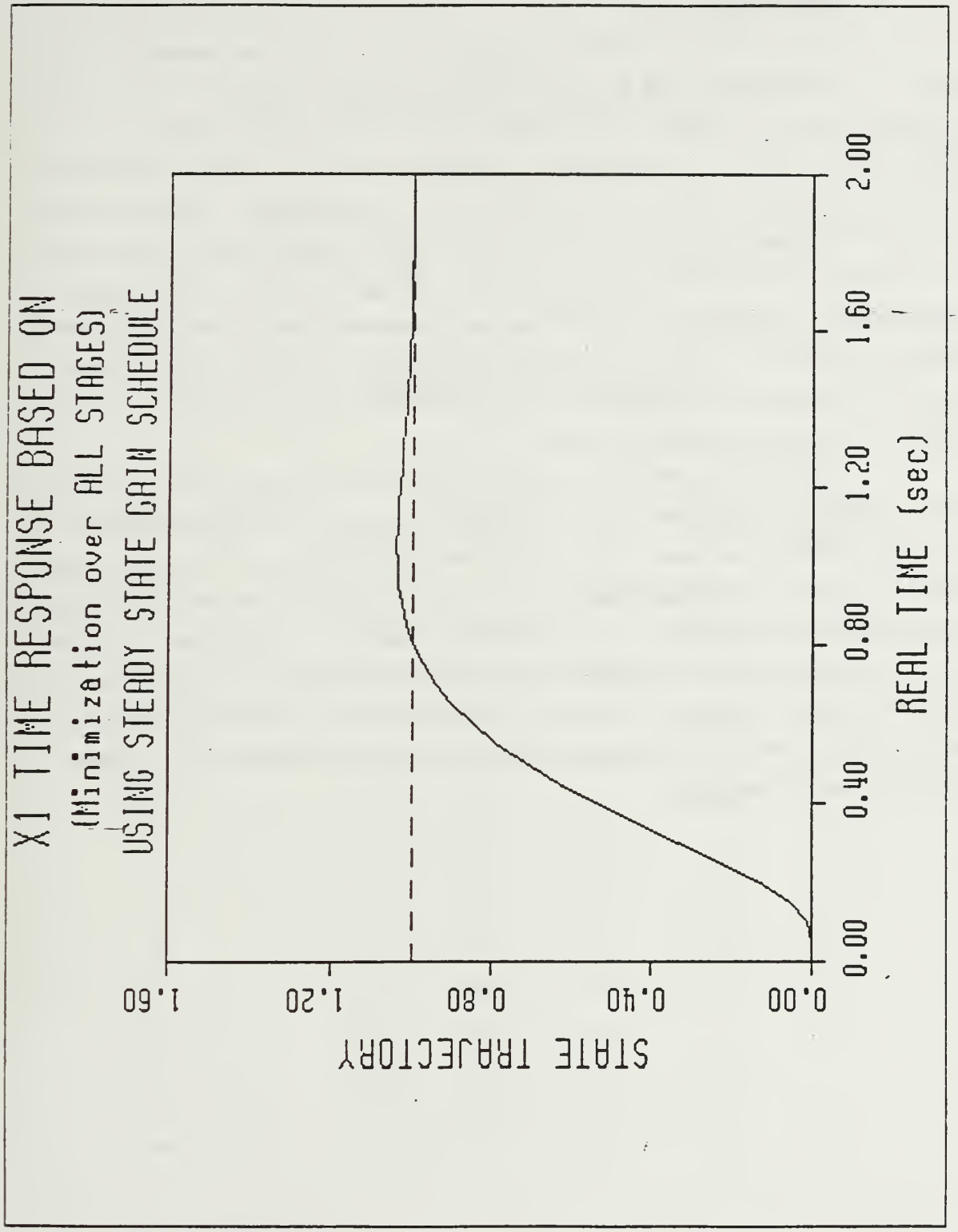

Figure 3.4 Roll Rate Time Response

b. Methodology

At this point, four different *groups* of cost functions are examined. For a given cost function group, the **H** and **Q** matrices are held constant while the control weighting factor, R, is varied within the range (0.01, 100). Approximately 15 *runs* are made for each cost function group with a different value of R inserted into the cost function for each run. After the steady state gains are determined for a given run, they are implemented into the control equation and a time response for the roll rate state, x_1, is obtained. The percent overshoot and 2% settling time are recorded for each x_1 time response. A summary of this information is presented for the four cost function groups in Tables 6, 7, 8, and 9. Following each of these four tables, there appears a graph of two time response parameters, percent overshoot and settling time, versus the value of the control weighting factor, R.

It is hardly necessary to include a time response graph for all of the 59 total runs examined. It is instructive, however, to compare the time responses for a selected set of cost functions. Three runs in each of the parameter summary tables, Tables 6 through 9, are flagged with asterisks. These flags indicate that the roll rate time response graph for that run is included subsequent to the applicable summary table. Keep in mind that the criteria for the roll rate step response is specified to be such that there is no overshoot and the 2% settling time is less than one second. A discussion of the results from the four cost function groups follows the last figure in this series of tables and graphs.

TABLE 6

ROLL RATE PARAMETERS FOR GROUP 1

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Sampling Interval T = 0.05 seconds

Run	Control Weight R	Steady State Gains f_1	f_2	f_3	Percent Overshoot	Settling Time (sec)
1	0.01	0.1727	-0.2944	-0.0892	4.03	1.31
2	0.03	0.1726	-0.2942	-0.0891	4.02	1.31
3	0.05	0.1725	-0.2940	-0.0890	4.02	1.31
4 *	0.10	0.1724	-0.2936	-0.0890	4.00	1.31
5	0.30	0.1717	-0.2920	-0.0884	3.95	1.31
6	0.50	0.1711	-0.2904	-0.0879	3.89	1.31
7	1.00	0.1696	-0.2866	-0.0866	3.74	1.30
8	3.00	0.1638	-0.2728	-0.0820	3.17	1.30
9	5.00	0.1587	-0.2610	-0.0780	2.68	1.29
10	7.00	0.1541	-0.2509	-0.0744	2.22	1.24
11	10.00	0.1480	-0.2380	-0.0697	1.65	0.87
12 **	20.00	0.1323	-0.2078	-0.0582	0.42	1.01
13	30.00	0.1209	-0.1886	-0.0504	0.00	1.17
14	50.00	0.1053	-0.1646	-0.0403	0.00	1.47
15 ***	100.00	0.0835	-0.1350	-0.0278	0.00	2.05

* See Figure 3.6.
** See Figure 3.7.
*** See Figure 3.8.

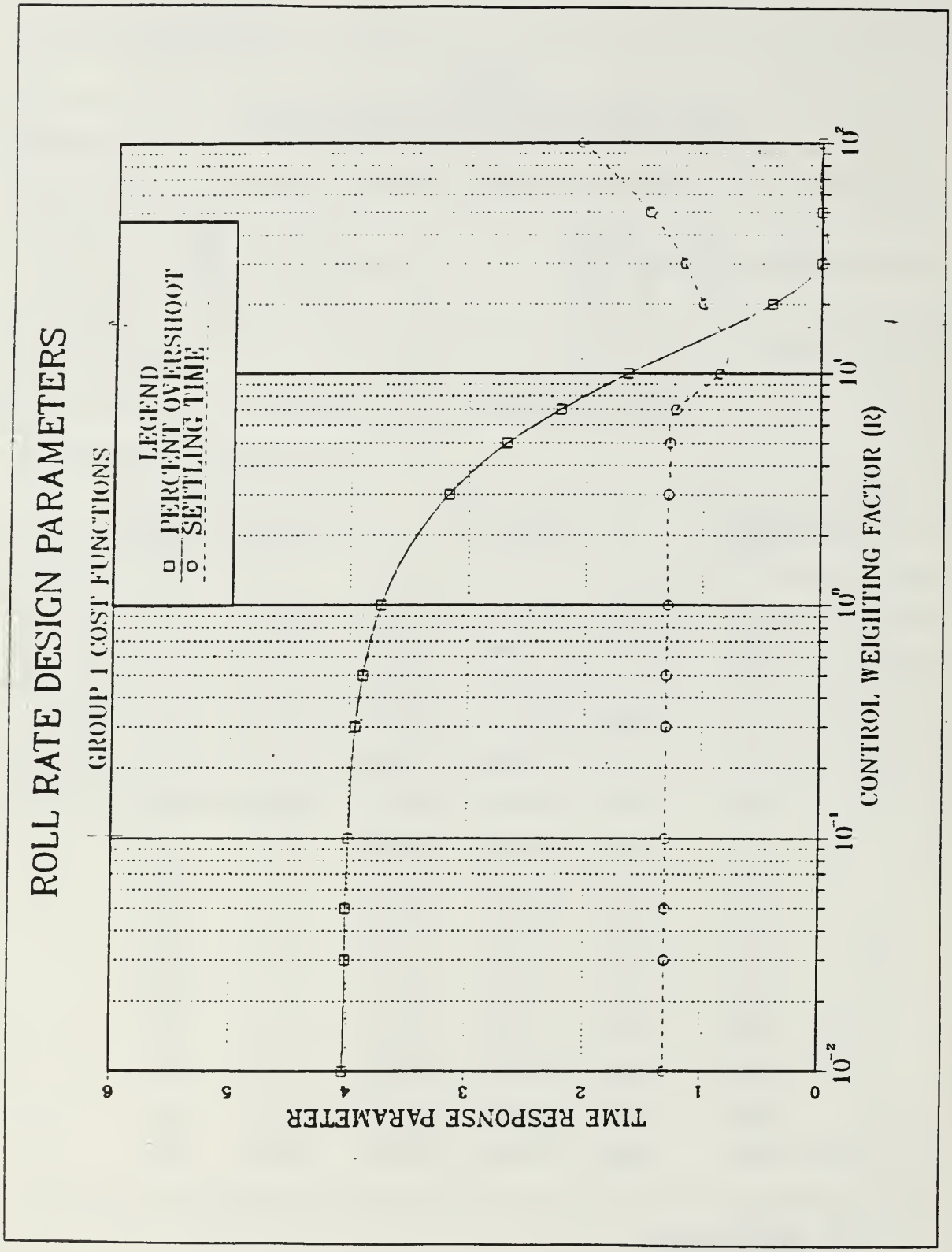

Figure 3.5 Group 1 Time Response Parameters

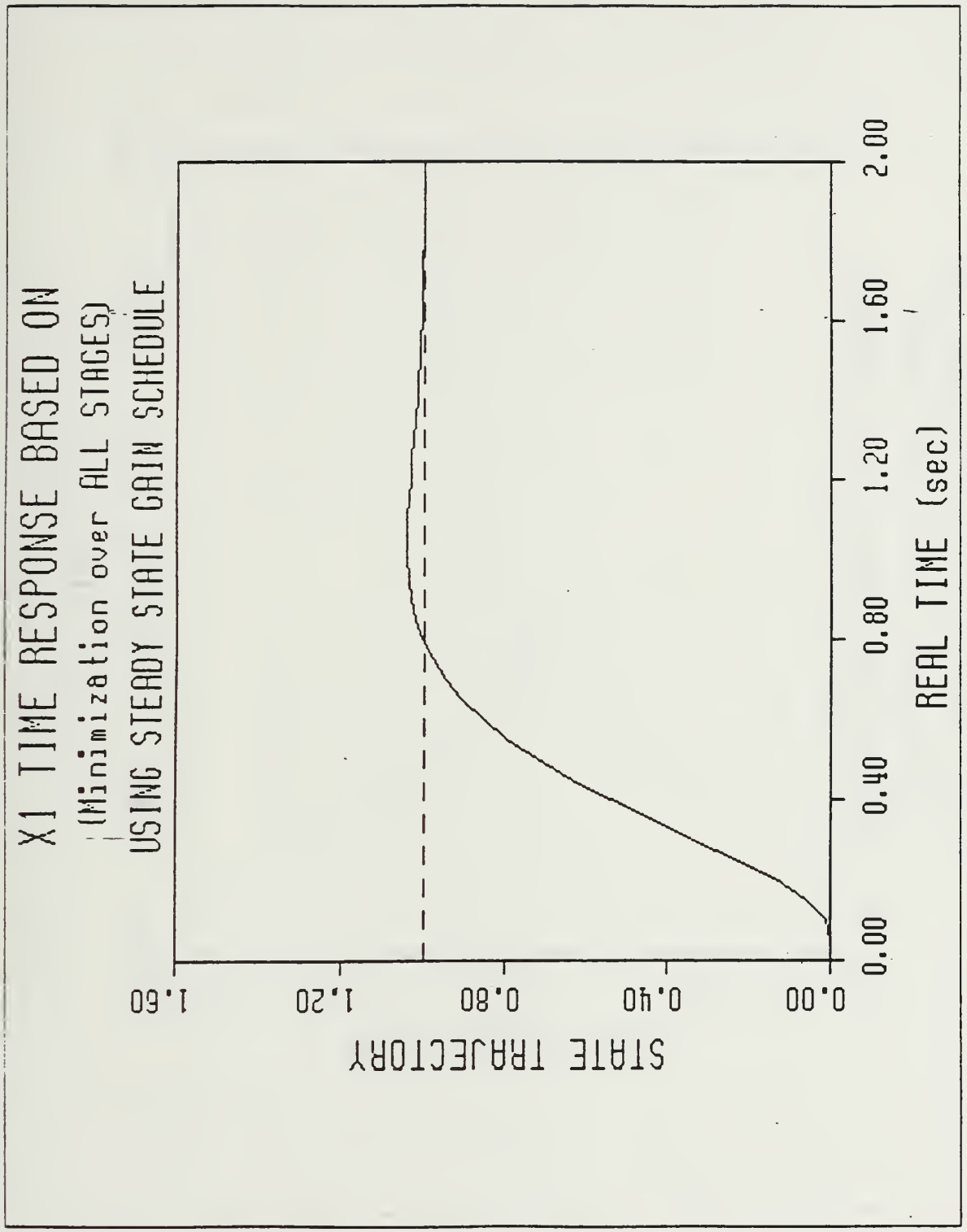

Figure 3.6 Roll Rate Time Response for Group 1 Run Number 4

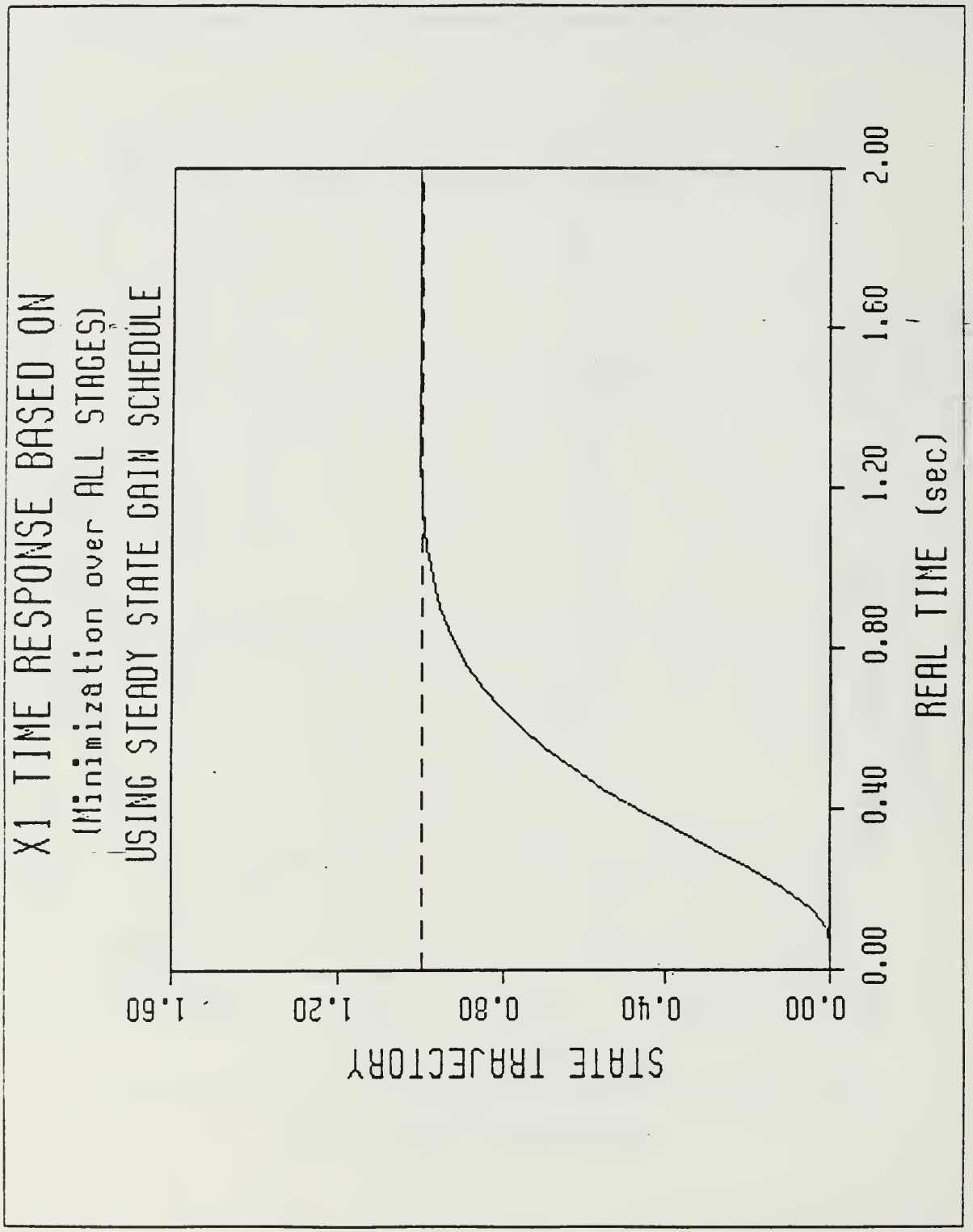

Figure 3.7 Roll Rate Time Response for Group 1 Run Number 12

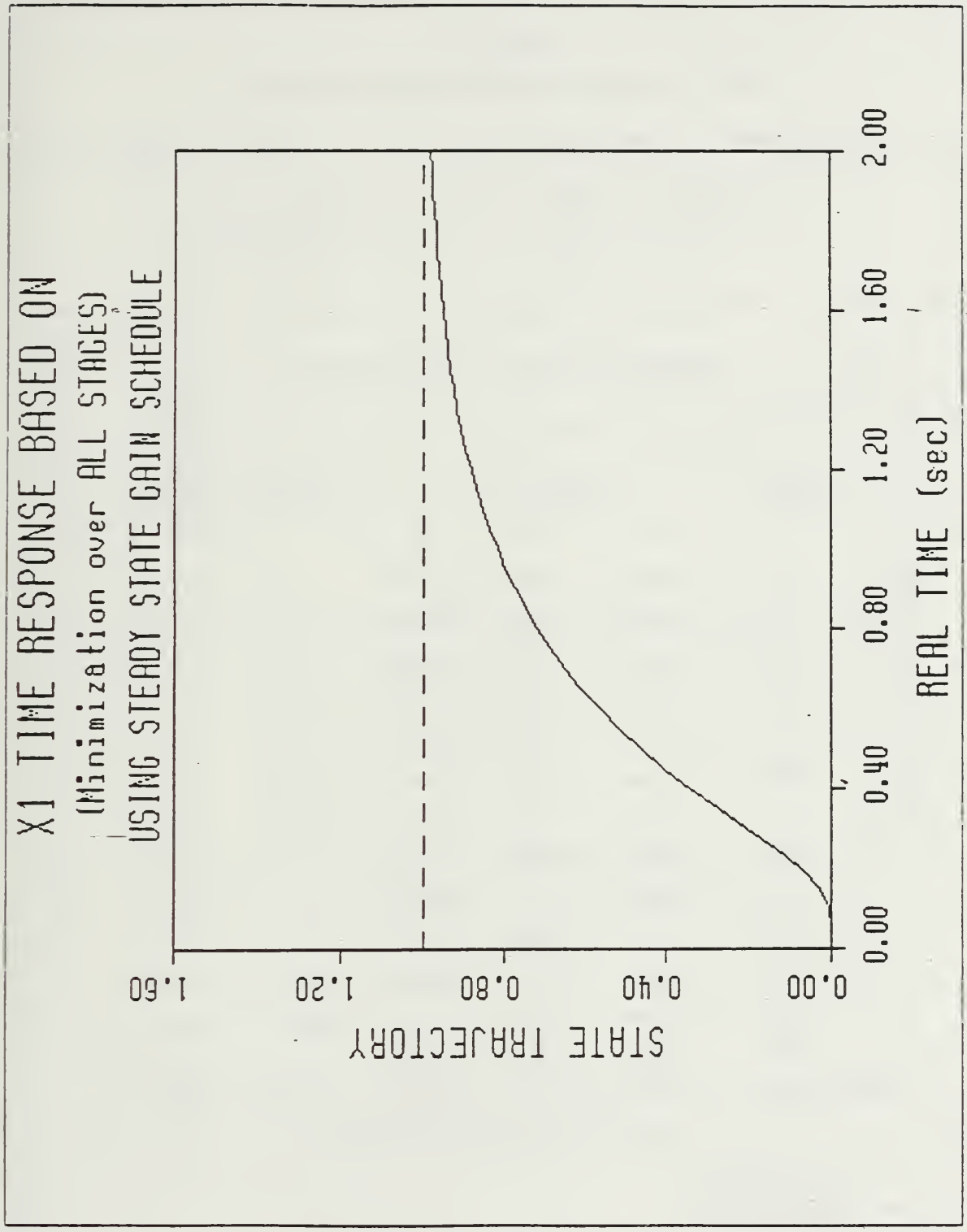

Figure 3.8 Roll Rate Time Response for Group 1 Run Number 15

TABLE 7

ROLL RATE PARAMETERS FOR GROUP 2

$$H = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

Sampling Interval T = 0.05 seconds

Run	Control Weight R	Steady State Gains f_1	f_2	f_3	Percent Overshoot	Settling Time (sec)
1	0.01	0.4803	-1.2269	-0.1073	4.37	0.79
2	0.03	0.4786	-1.2218	-0.1068	4.35	0.79
3	0.05	0.4769	-1.2168	-0.1063	4.33	0.79
4 *	0.10	0.4728	-1.2047	-0.1052	4.28	0.79
5	0.30	0.4576	-1.1598	-0.1009	4.08	0.79
6	0.50	0.4441	-1.1202	-0.0971	3.88	0.79
7	1.00	0.4159	-1.0381	-0.0893	3.49	0.79
8	3.00	0.3442	-0.8374	-0.0700	2.12	0.73
9	5.00	0.3024	-0.7251	-0.0593	1.12	0.57
10	7.00	0.2737	-0.6504	-0.0522	0.44	0.62
11 **	10.00	0.2435	-0.5735	-0.0450	0.00	0.70
12	30.00	0.1596	-0.3702	-0.0268	0.00	1.16
13	50.00	0.1281	-0.2969	-0.0206	0.00	1.46
14 ***	100.00	0.0937	-0.2175	-0.0144	0.00	2.00

* See Figure 3.10.
** See Figure 3.11.
*** See Figure 3.12.

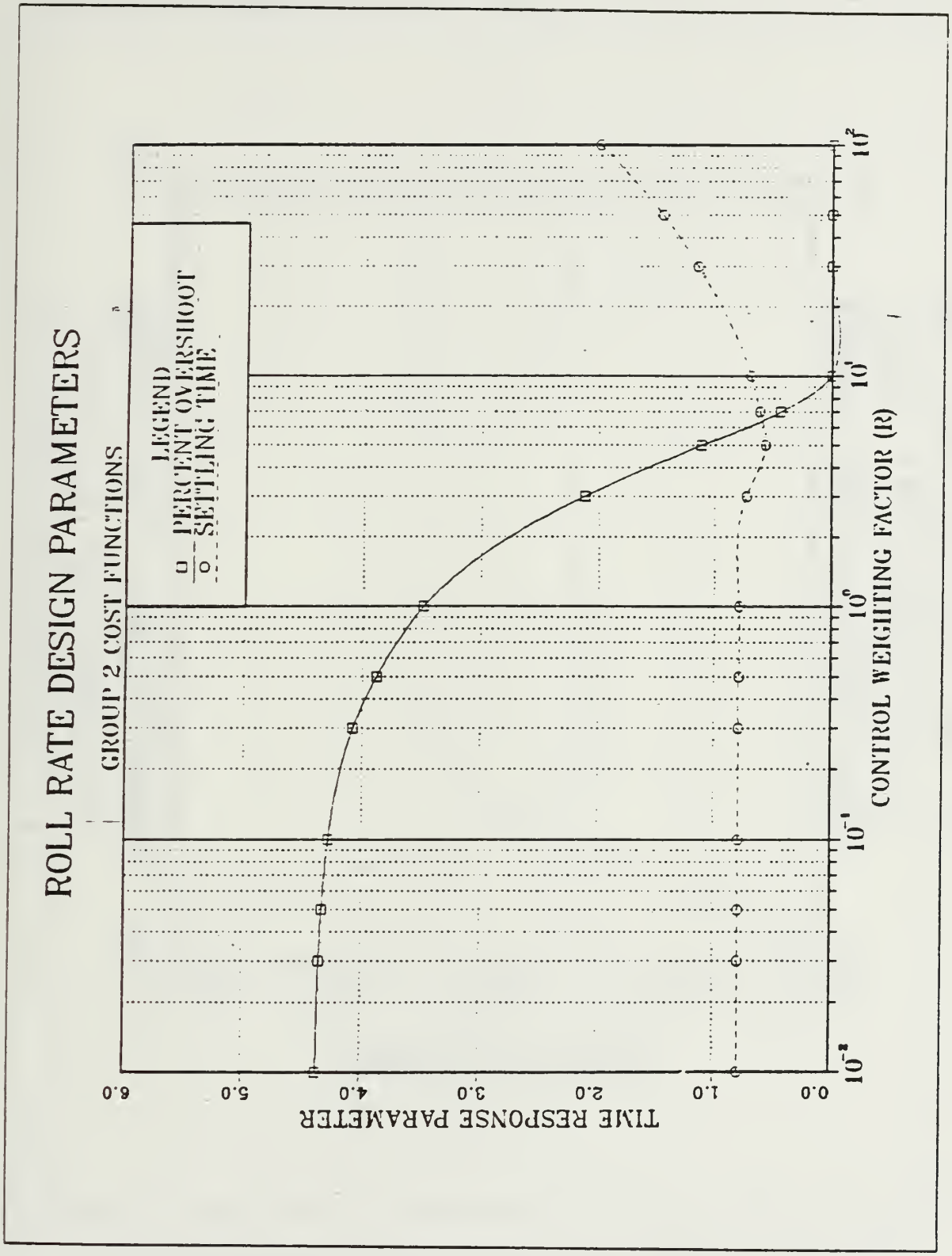

Figure 3.9 Group 2 Time Response Parameters

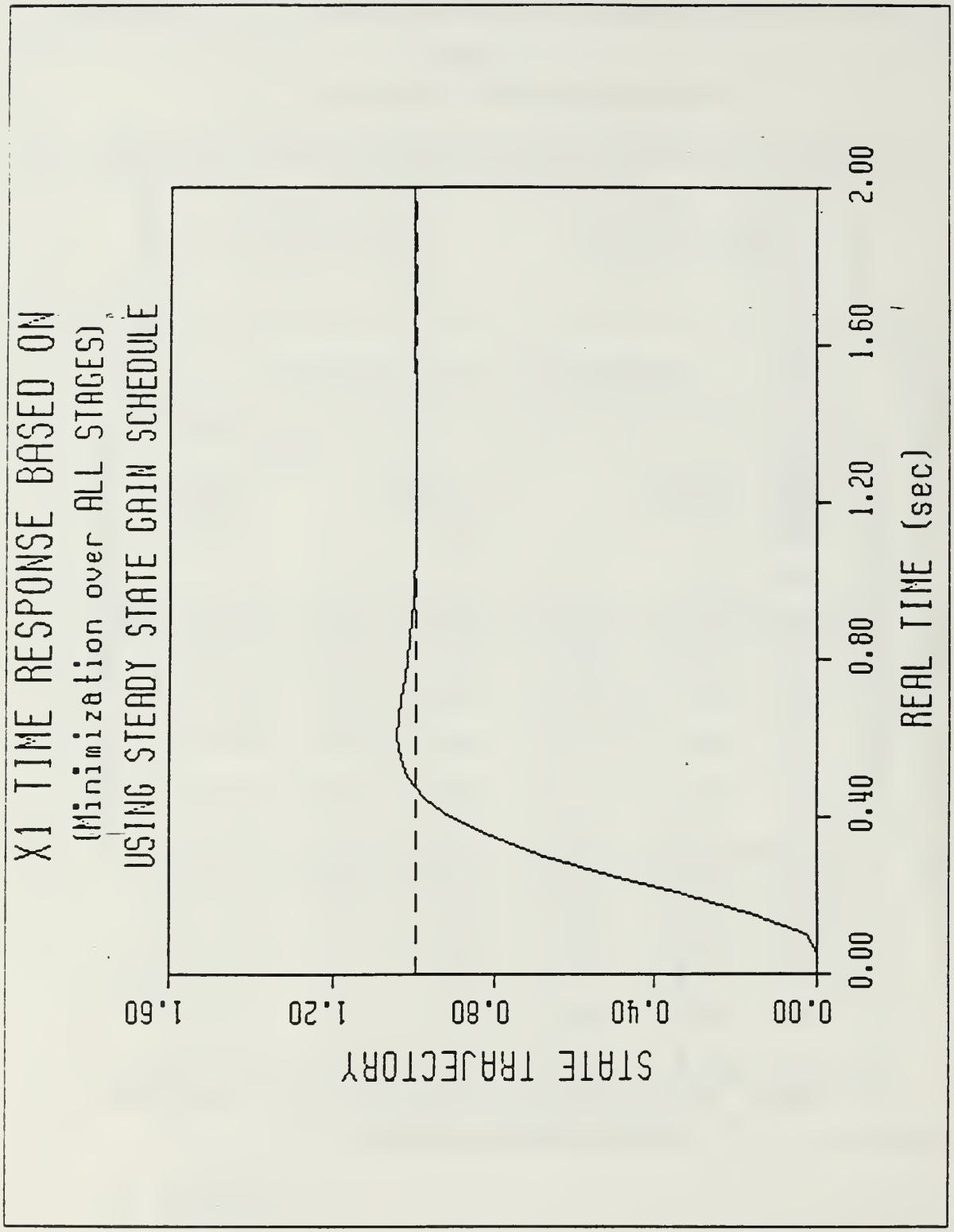

Figure 3.10 Roll Rate Time Response for Group 2 Run Number 4

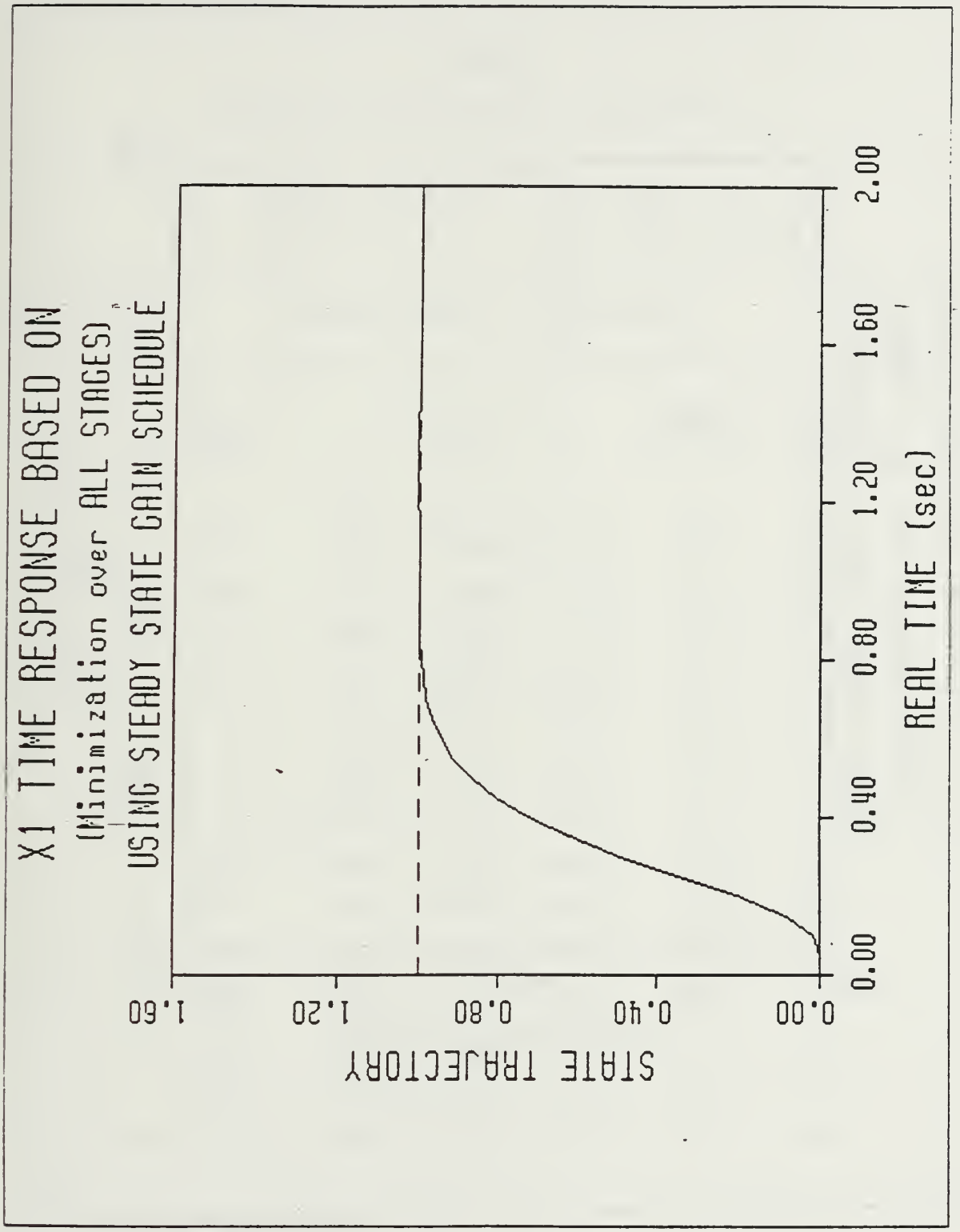

Figure 3.11 Roll Rate Time Response for Group 2 Run Number 11

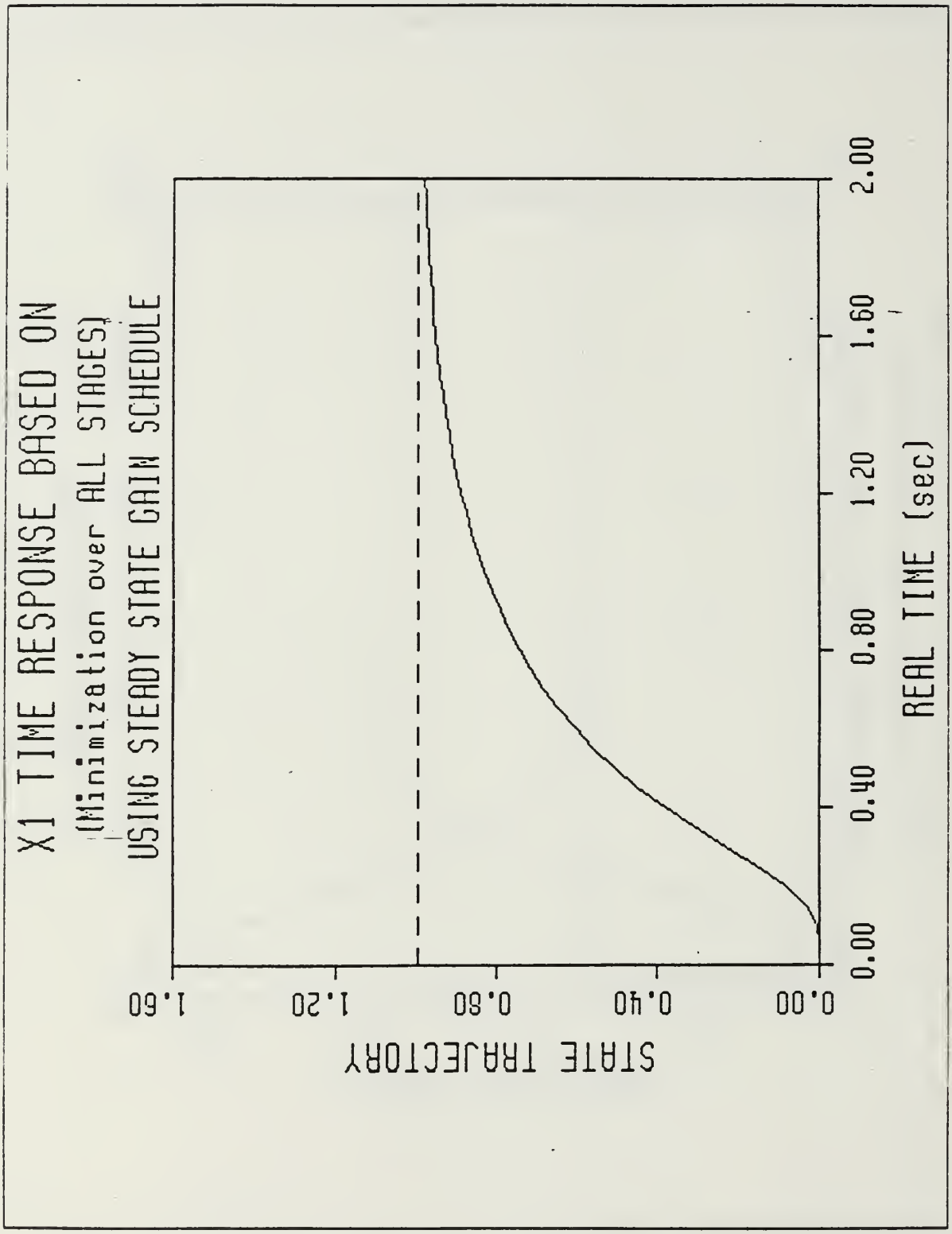

Figure 3.12 Roll Rate Time Response for Group 2 Run Number 14

TABLE 8

ROLL RATE PARAMETERS FOR GROUP 3

$$H = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

Sampling Interval T = 0.05 seconds

Run	Control Weight R	Steady State Gains f_1	f_2	f_3	Percent Overshoot	Settling Time (sec)
1	0.01	1.1983	-2.6905	-0.1332	4.53	0.48
2	0.03	1.1622	-2.6141	-0.1296	4.54	0.49
3	0.05	1.1300	-2.5460	-0.1264	4.58	0.49
4 *	0.10	1.0625	-2.4028	-0.1196	4.65	0.49
5	0.30	0.8917	-2.0369	-0.1023	4.63	0.51
6	0.50	0.7917	-1.8200	-0.0919	4.29	0.52
7	1.00	0.6497	-1.5079	-0.0770	3.64	0.53
8	1.50	0.5689	-1.3280	-0.0683	2.87	0.54
9	2.00	0.5144	-1.2053	-0.0623	2.33	0.53
10	3.00	0.4426	-1.0425	-0.0543	1.28	0.43
11	5.00	0.3621	-0.8576	-0.0452	0.00	0.50
12	10.00	0.2712	-0.6458	-0.0345	0.00	0.71
13	15.00	0.2273	-0.5427	-0.0292	0.00	0.87
14 **	20.00	0.2001	-0.4782	-0.0258	0.00	0.97
15	30.00	0.1663	-0.3986	-0.0217	0.00	1.16
16	50.00	0.1316	-0.3153	-0.0172	0.00	1.46
17 ***	100.00	0.0950	-0.2277	-0.0126	0.00	2.00

* See Figure 3.14.
** See Figure 3.15.
*** See Figure 3.16.

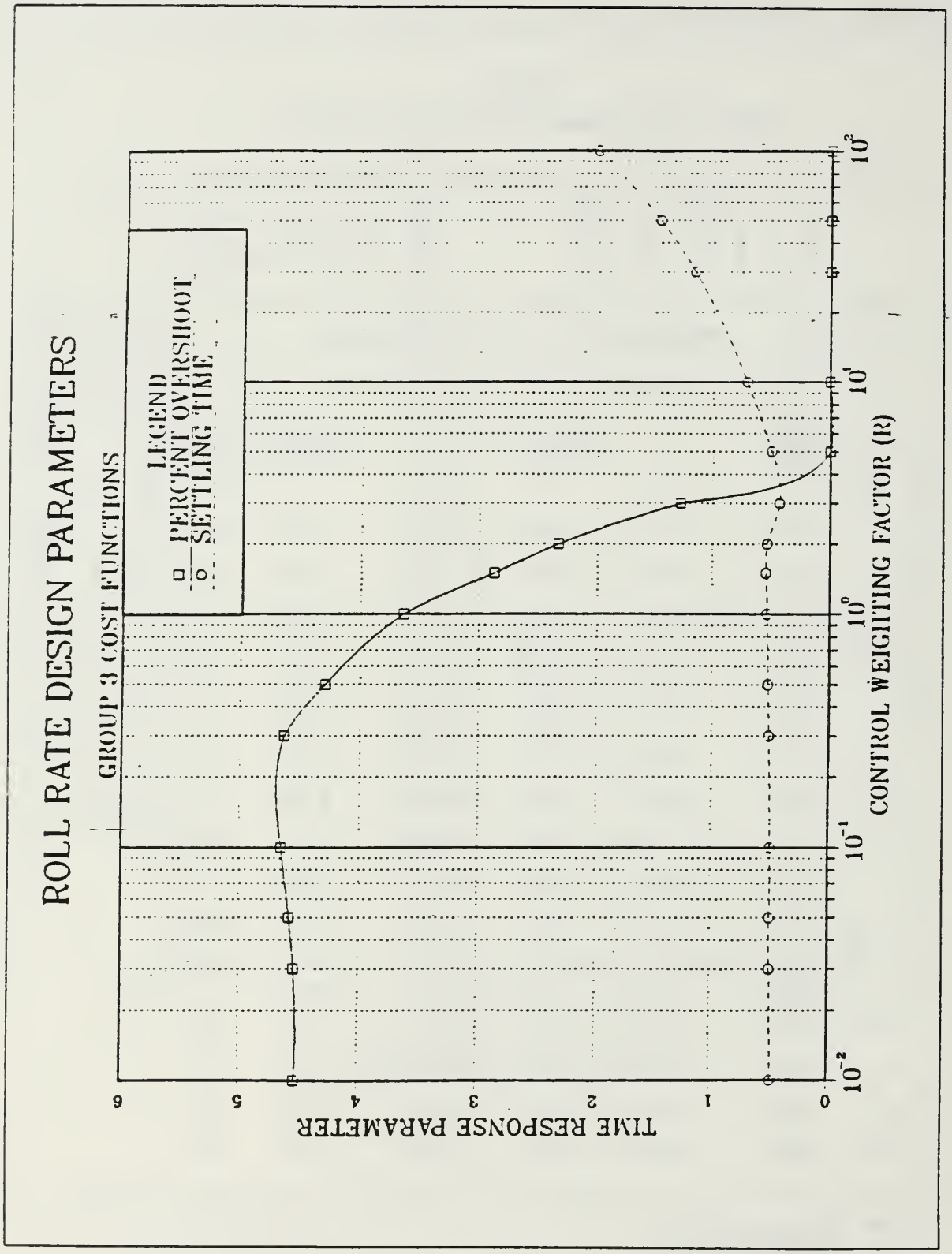

Figure 3.13 Group 3 Time Response Parameters

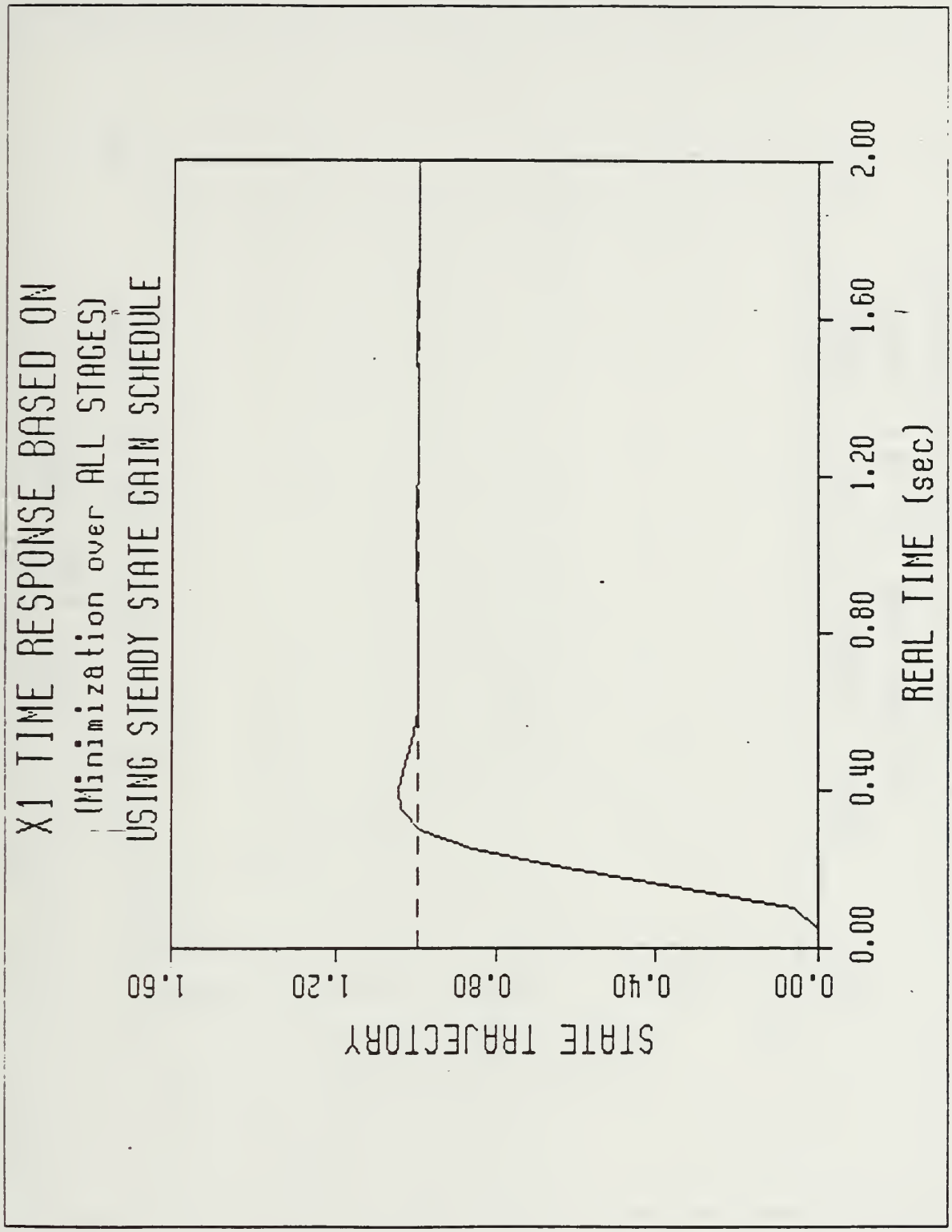

Figure 3.14 Roll Rate Time Response for Group 3 Run Number 4

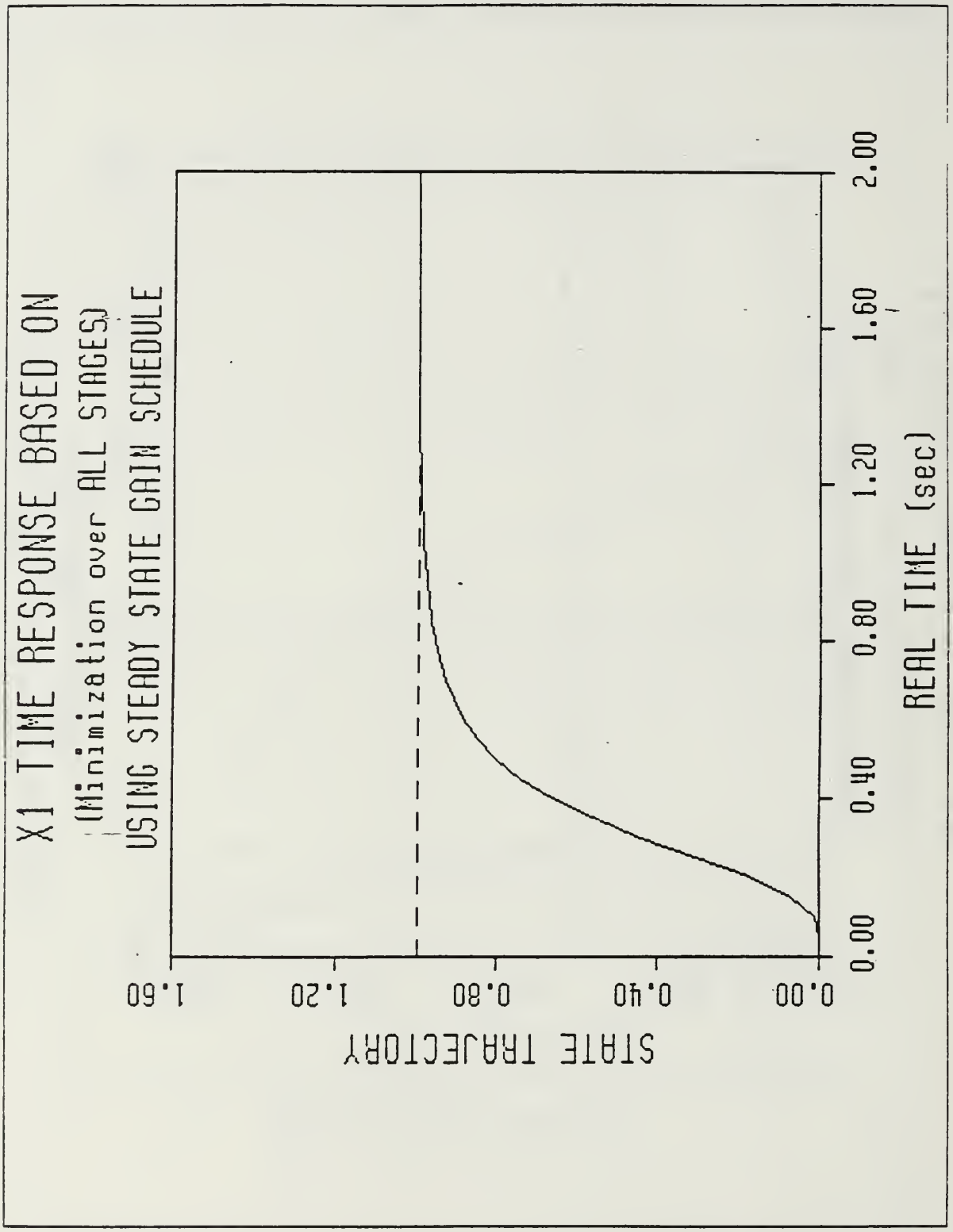

Figure 3.15 Roll Rate Time Response for Group 3 Run Number 14

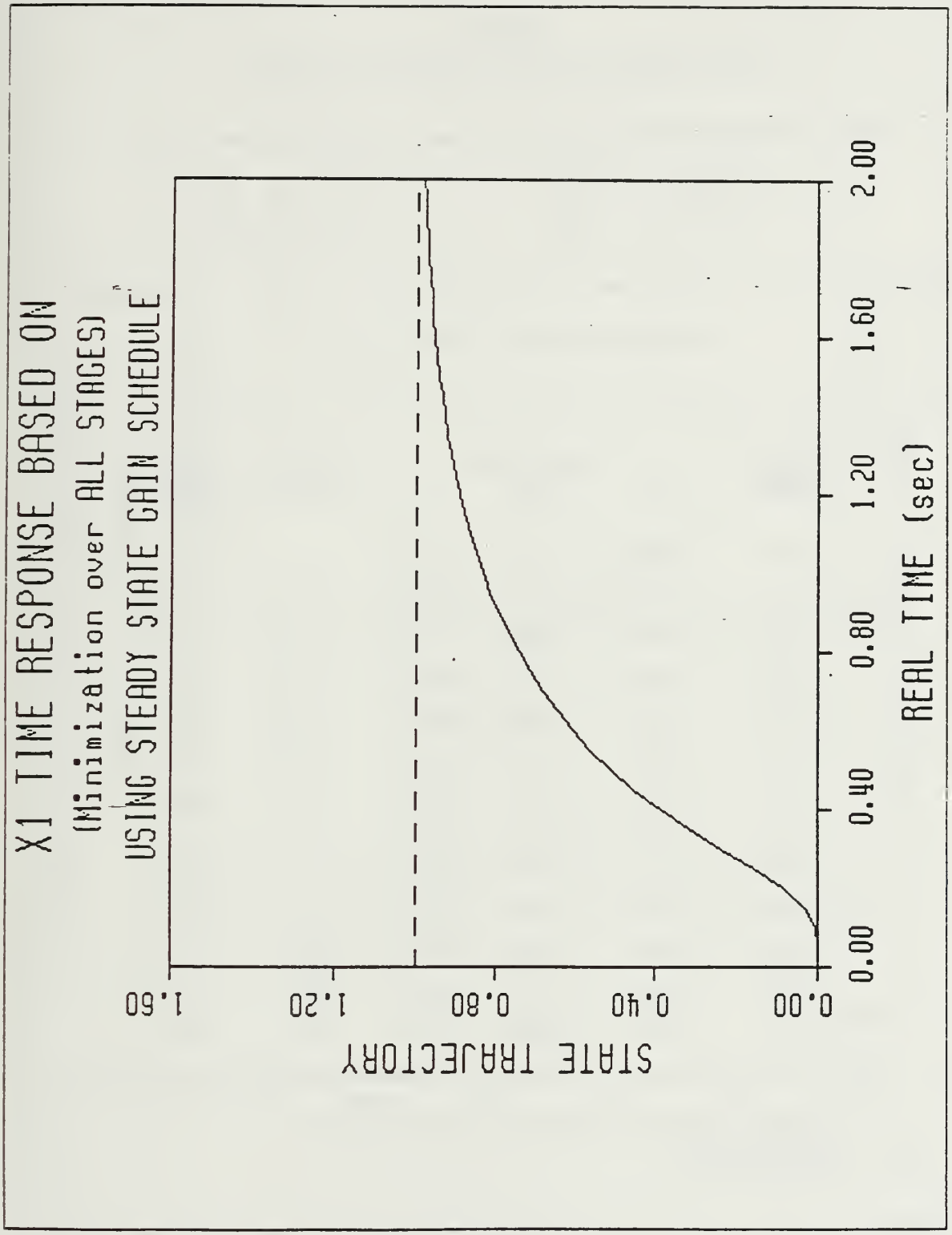

Figure 3.16 Roll Rate Time Response for Group 3 Run Number 17

TABLE 9

ROLL RATE PARAMETERS FOR GROUP 4

$$H = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Sampling Interval T = 0.05 seconds

Run	Control Weight R	Steady State Gains f_1	f_2	f_3	Percent Overshoot	Settling Time (sec)
1	0.01	3.1663	-5.4007	-0.1713	9.57	0.27
2	0.03	2.3689	-4.3357	-0.1480	8.82	0.31
3	0.05	2.0386	-3.8586	-0.1366	8.67	0.33
4 *	0.10	1.6396	-3.2461	-0.1209	7.37	0.36
5	0.30	1.1264	-2.3821	-0.0962	6.22	0.42
6	0.50	0.9350	-2.0316	-0.0852	5.96	0.44
7	1.00	0.7182	-1.6110	-0.0709	4.51	0.48
8	3.00	0.4613	-1.0739	-0.0506	1.33	0.41
9	5.00	0.3718	-0.8759	-0.0425	0.00	0.49
10 **	10.00	0.2750	-0.6551	-0.0329	0.00	0.72
11	30.00	0.1674	-0.4020	-0.0210	0.00	1.16
12	50.00	0.1320	-0.3175	-0.0168	0.00	1.46
13 ***	100.00	0.0951	-0.2289	-0.0123	0.00	2.00

* See Figure 3.18.
** See Figure 3.19.
*** See Figure 3.20.

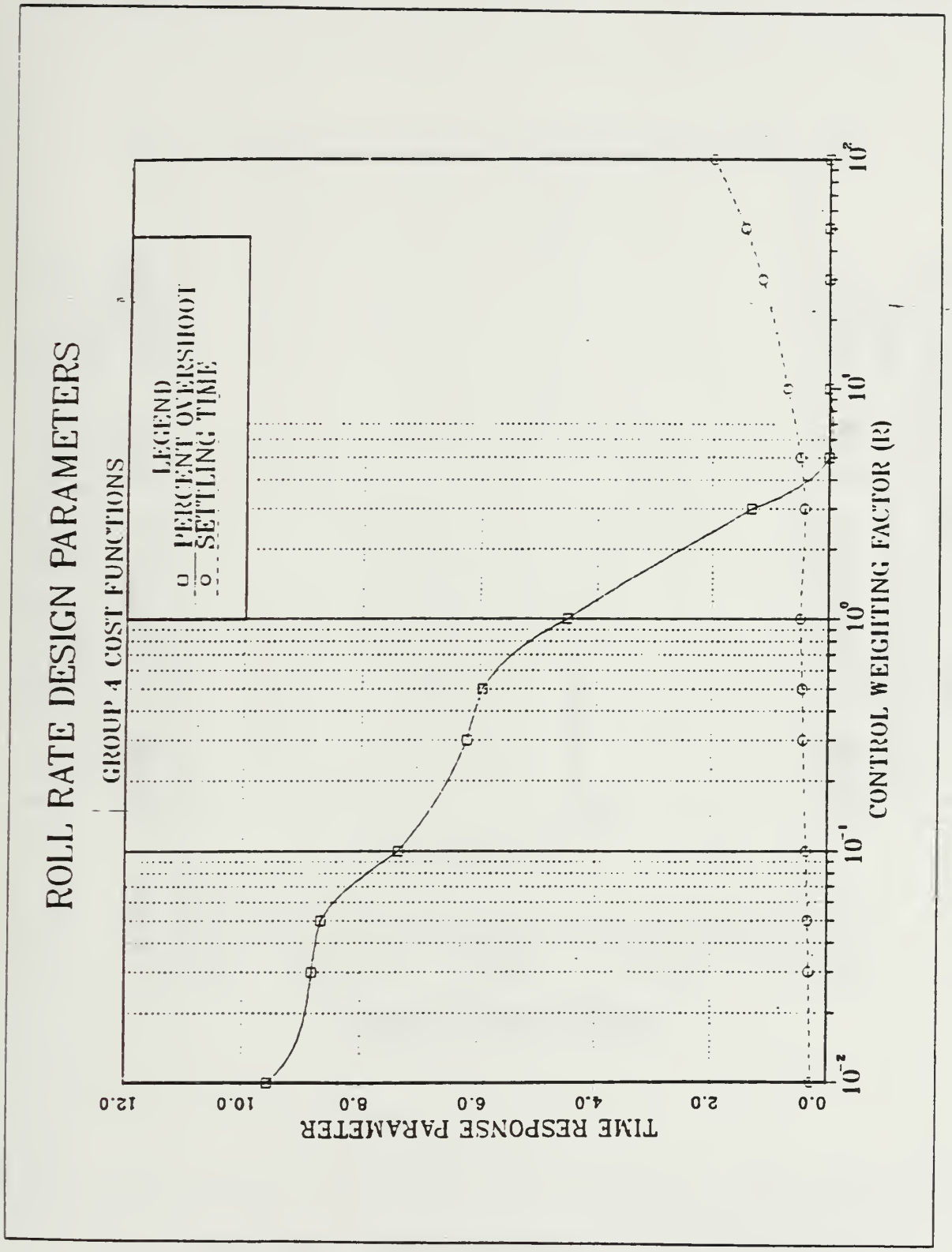

Figure 3.17 Group 4 Time Response Parameters

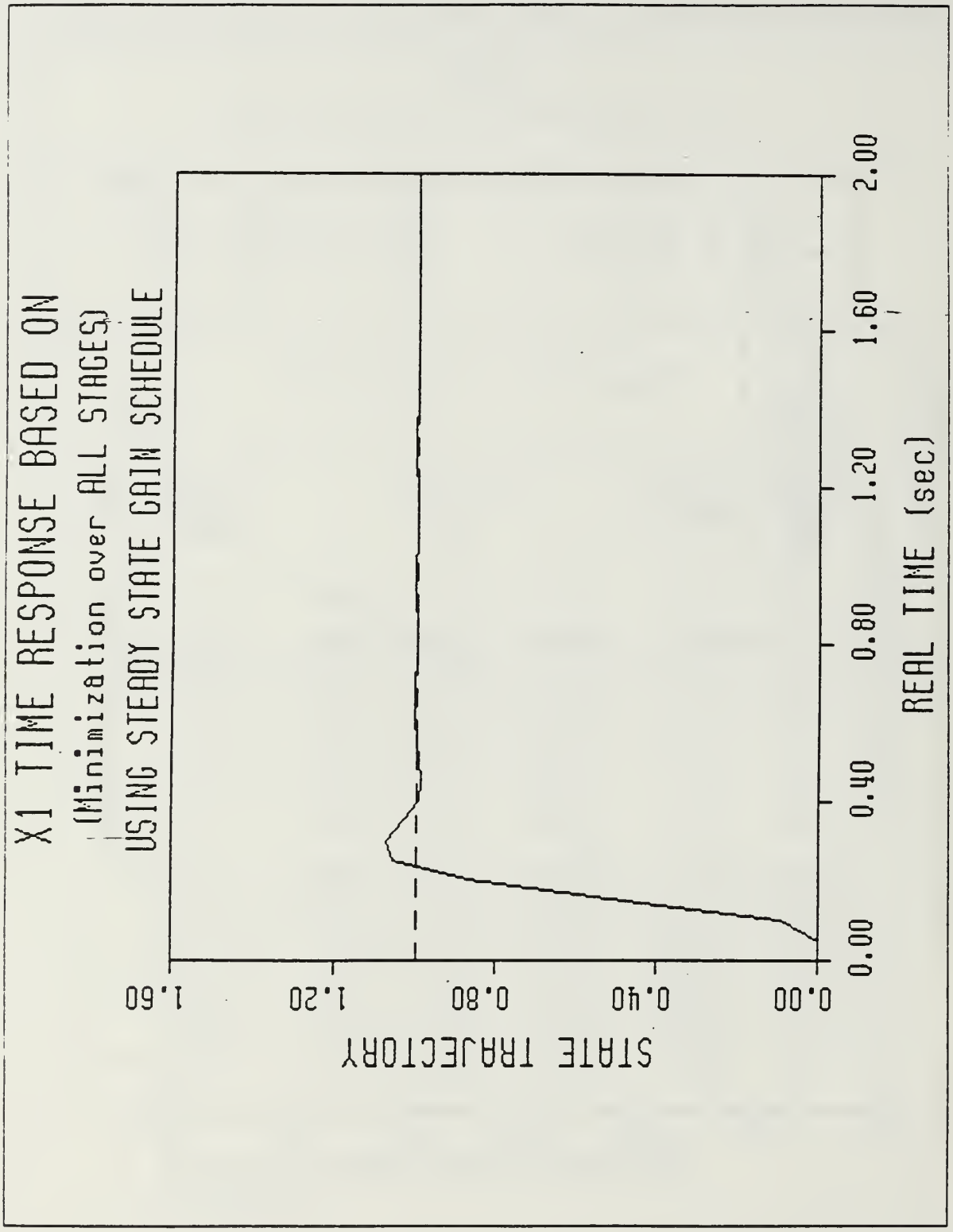

Figure 3.18 Roll Rate Time Response for Group 4 Run Number 4

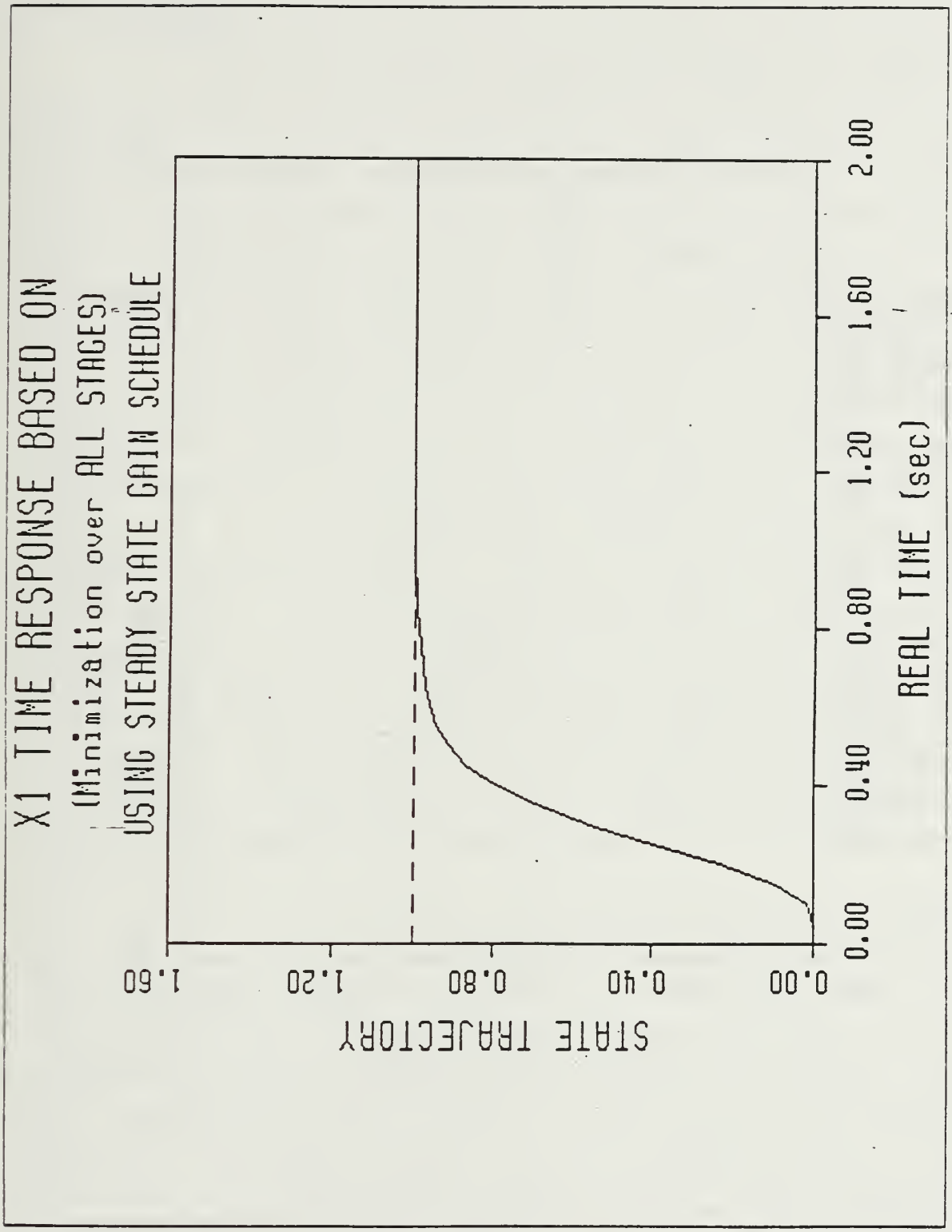

Figure 3.19 Roll Rate Time Response for Group 4 Run Number 10

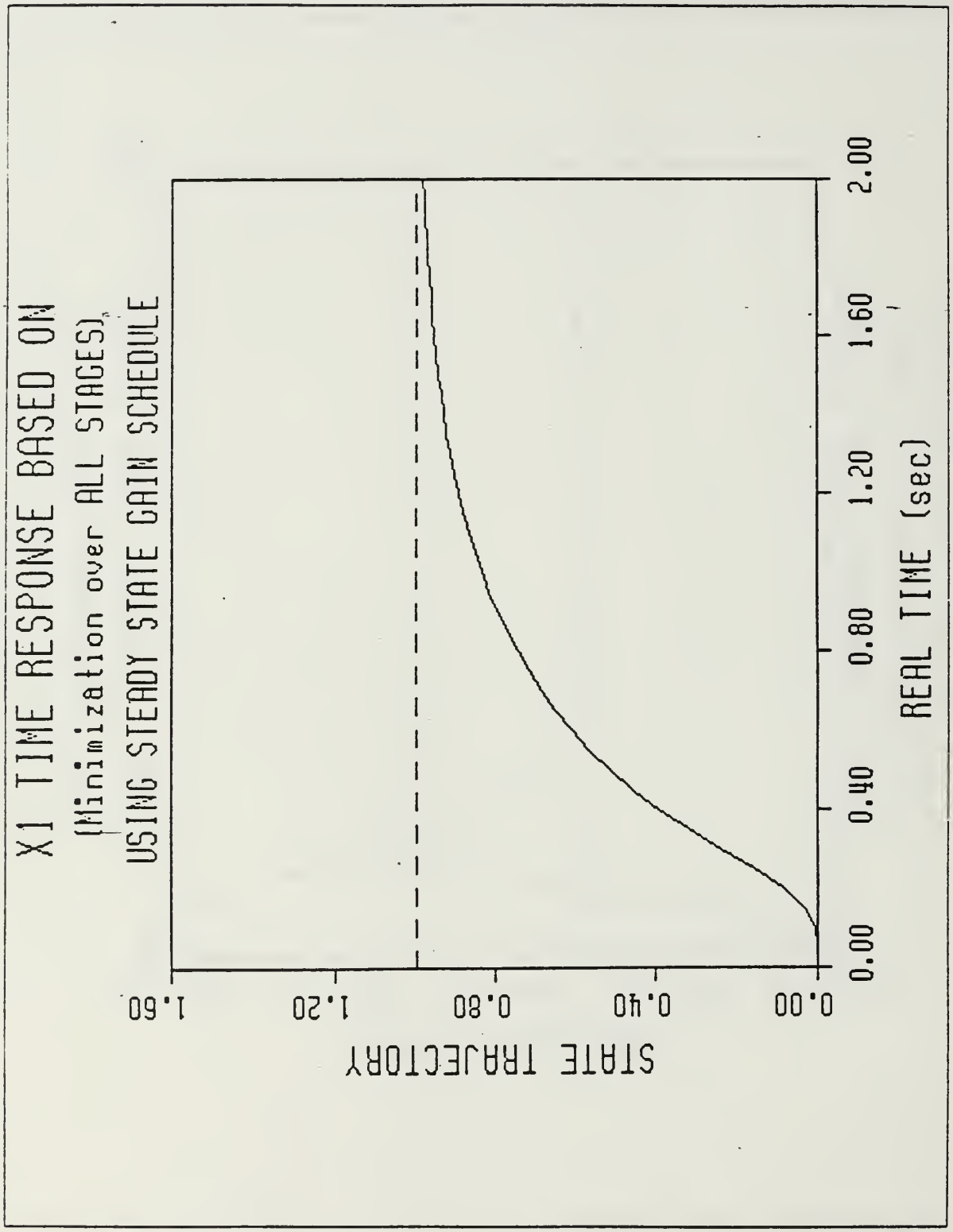

Figure 3.20 Roll Rate Time Response for Group 4 Run Number 13

c. Results

(1) *Group 1 Cost Functions.* All three states are weighted with a value of unity for the Group 1 cost functions. This implies that the designer is attempting to minimize the error in all states with equal emphasis. Pursuit of this group of cost functions is made in order to determine general patterns of cause and effect. For example, it is apparent in Table 6 that the control weighting factor, R, significantly affects the magnitude of the steady state optimal feedback gain vector, $\mathbf{F}_{ss}$. By increasing the penalty on the control effort, the magnitudes of the feedback gains are reduced. Thus, if the control system is physically limited to some maximum value of control effort, then the R term is the logical parameter to adjust. The percent overshoot and settling time data from Table 6 indicates that R directly affects the time response as well. From Figure 3.5, note that the value of R has negligible effect on the time response parameters for any value of R less than unity. Refer to Figure 3.6 in which R = 0.1. As the control weighting factor increases above unity, however, the time response is dramatically affected. For values of R greater than 30, the time response exhibits no overshoot and the settling time appears to lengthen without bound as the system becomes increasingly slow. Refer to Figure 3.8 in which R = 100. Also notice in Figure 3.5 that there is no cost function in Group 1 that yields an acceptable time response which satisfies both of the roll rate criteria. The cost function in Group 1 which yields a time response *closest* to the design specifications is run number 12 shown in Figure 3.7. This run is subsequently used as a basis for comparison of the time responses generated by the other three groups of cost functions.

(2) *Group 2 Cost Functions.* Because acceptable results are not obtained from the Group 1 cost functions, it is decided to place increased emphasis on the error in the x_1 state while reducing the emphasis on the error in the x_2 and x_3 states. This tactic is allowable because the maximum absolute values of the x_2 and x_3 states are significantly less than the constraints for the δ_a and $\dot{\delta}_a$ servo states listed in Table 4. Table 7 summarizes the data for Group 2. Figure 3.9 evidences the relationship between the x_1 time response parameters and the control weighting factor, R. Notice in this figure that an acceptable time response is expected for any Group 2 cost function in which $10 \leq R \leq 20$. Figure 3.11 shows the x_1 time response for run number 11 in which R = 10. This time response meets the required specifications for roll rate. Note, however, that the gains for this run are approximately 80% higher, on

the average, than the gains for the best run, number 12, in Group 1. In order to reduce the gains so that only a small control effort is demanded, there is more work yet to be done.

(3) *Group 3 Cost Functions.* Making the penalty on the x_1 error state 10 times greater than the penalty on the x_2 and x_3 error states in the Group 2 cost functions appears to be a reasonable mechanism for obtaining an adequate time response. In an effort to reduce the magnitude of the control effort, the ratios of h_{11}/h_{22}, h_{11}/h_{33}, q_{11}/q_{22}, and q_{11}/q_{33} are increased to 100 for the Group 3 cost functions. Table 8 and Figure 3.13 present the data for this group. The time responses obtained for this group are very similar to those obtained for Groups 1 and 2. Notice in Figure 3.13 that the overshoot is zero for all cases in which $R \geq 5$. In addition, the settling time is less than one second when $R \leq 20$. The steady state gains for run number 14 average only 42% greater than $\mathbf{F}_{ss}$ for run number 12 in Group 1. Thus the hypothesis tested in the Group 3 cost functions is validated.

(4) *Group 4 Cost Functions.* The cost functions in Group 4 penalize errors *only* in the x_1 state and the control effort. That all other elements of **H** and **Q** are zero implies that no penalty is assessed against deviations in the x_2 and x_3 states. Table 9 and Figure 3.17 present the data for this group. Run number 10 is deemed to be the most acceptable time response and is shown in Figure 3.19. While this design satisfies the design criteria, note that the steady state control gains average 93% greater than the most acceptable run in Group 1. This is the greatest increase in control gains that is observed. Also notice that the percent overshoot curve in Figure 3.17 increases upwards of 9%. This is much greater than the maximum overshoot of 4.65% observed in Groups 1, 2, and 3. For these two reasons, it is determined that the cost functions tested in Group 4 do not need to be further pursued.

(5) *Summary.* Figures 3.21 and 3.22 summarize the information contained in Tables 6, 7, 8, and 9. It is interesting to note in Figure 3.21 that the first three groups of cost functions yield suprisingly similar curves for the percent overshoot of the roll rate system. That the Group 4 cost functions produce a much more erratic curve is attributed to the fact that no weight is placed on the x_2 or x_3 states in this group. The roll rate settling times in Figure 3.22 exhibit similar patterns for all four groups of cost functions. Note that in all cases there appears to be a *minimum* settling time possible when $2 \leq R \leq 10$. For values of $R > 10$, the large emphasis on control effort produces small steady state gains which in turn yield a slow system.

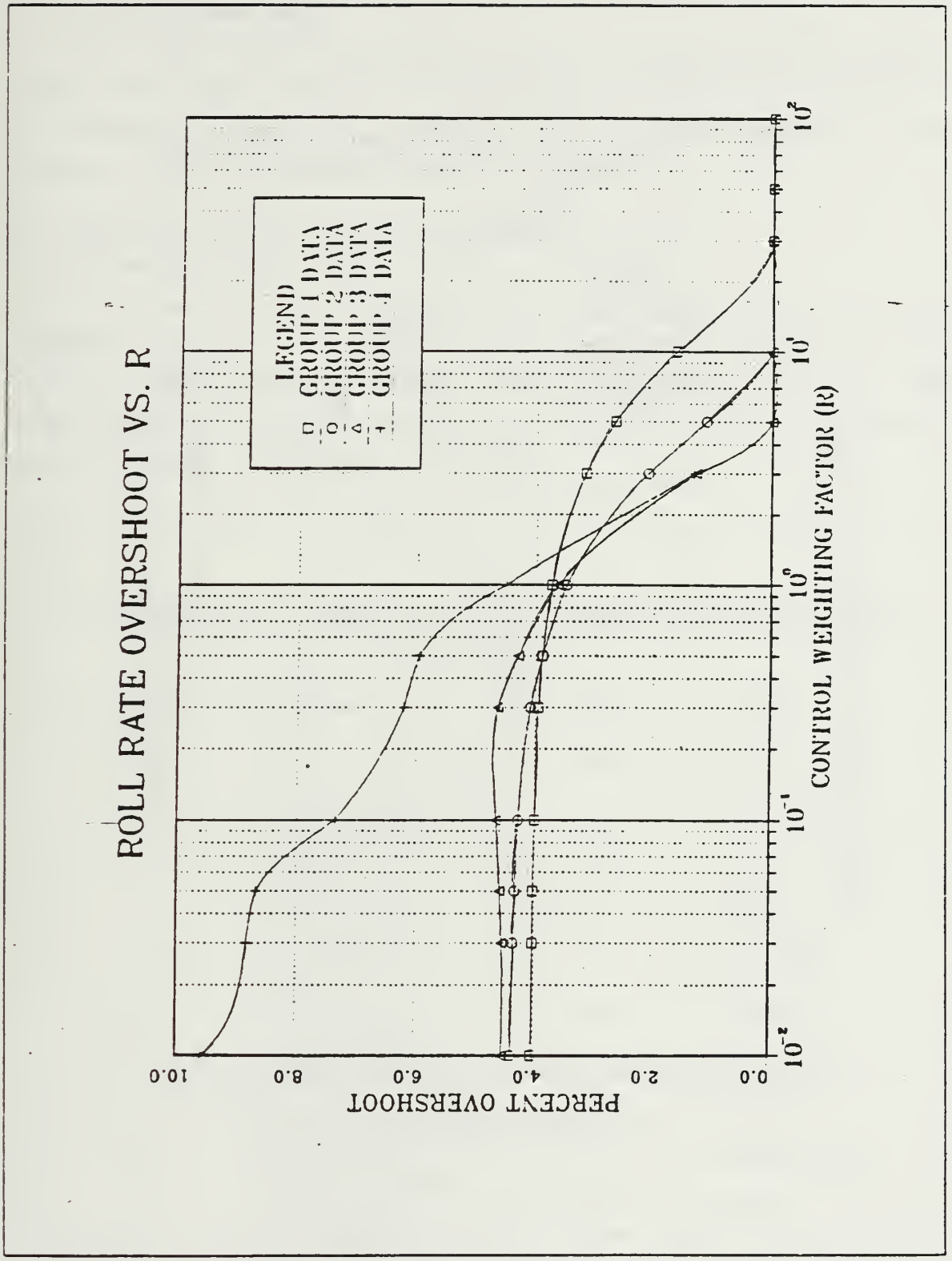

Figure 3.21 Percent Overshoot Curves For Groups 1, 2, 3, and 4

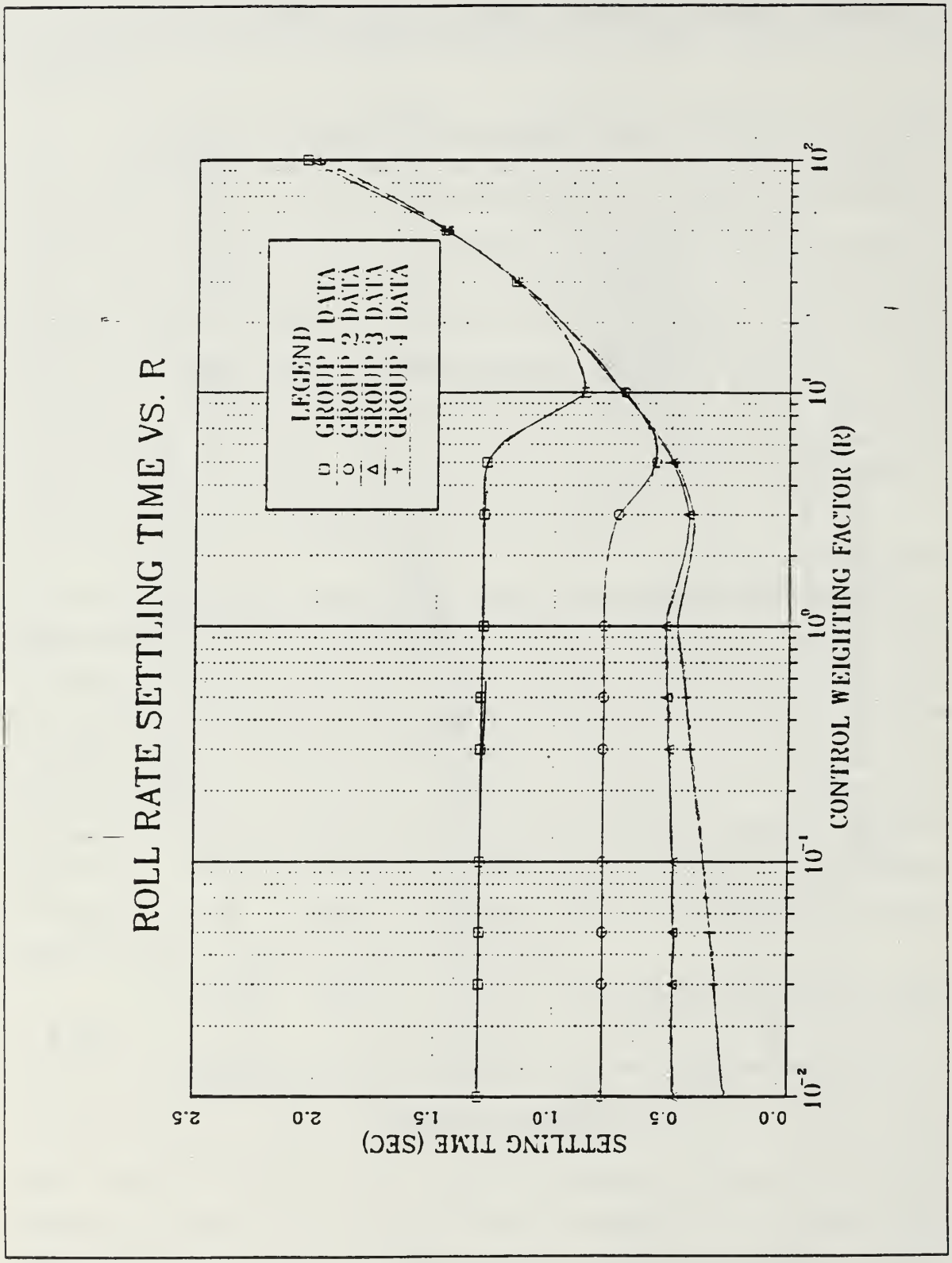

Figure 3.22 Settling Time Curves For Groups 1, 2, 3, and 4

d. The Final Design

Based on the time response specifications and on the desire to design a control system which implements minimal steady state gains, it is decided that run number 14 in Group 3 is the best solution for a roll rate controller. The time response for this set of parameters appears in Figure 3.15.

C. AROD ALTITUDE RATE CONTROLLER

1. The Altitude Rate System

Because the primary flight mode for AROD is low altitude hover, it is important that there be a reliable control system to maintain the vehicle's vertical position relative to the earth's surface. The throttle on AROD's two cycle engine provides the mechanism for altitude rate control. Table 10 defines the terms which are involved in the altitude rate equations of motion.

TABLE 10
VARIABLE DEFINITIONS FOR AROD ALTITUDE RATE EQUATIONS OF MOTION

Variable	Definition	Value	Units
$\dot{h}$	Vehicle Altitude Rate	TBD	feet/second
C_h	Engine Thrust to RPM Dynamic Coefficient	0.0865	ft/seconds^2/rad
δ_s	Change in Engine Speed	TBD	RPM
τ_e	Engine Lag Time Constant	0.5	seconds
K_e	Engine Scale Factor	837.8	rad/sec/rad
δ_t	Throttle Servo Deflection Angle	$\leq 30°$	radians
$\dot{\delta}_t$	Throttle Servo Deflection Velocity	$\leq 50°$/sec	radians/second
ζ	Throttle Servo Damping Coefficient	0.707	unitless
ω	Throttle Servo Natural Frequency	12.57	radians/second
u_t	Control Input to Throttle Servo	TBD	volts

By commanding a desired altitude rate, $\dot{h}_c$, the pilot sets in motion the following sequence of events :

1. A throttle servo control signal, u_t, is generated within the controller.
2. The throttle servo position is adjusted.
3. The actuator position commands a specific engine speed.
4. A change in engine RPM causes a change in the vehicle altitude rate.

The rate of change, $\ddot{h}$, of the vehicle's altitude rate, $\dot{h}$, is proportional to the change in engine RPM as shown in Equation 3.8.

$$\ddot{h} = C_h \delta_s \tag{3.8}$$

where the dynamic constant, C_h, is experimentally determined in wind tunnel tests. The engine is modelled as a first order lag system according to Equation 3.9.

$$\dot{\delta}_s = (-1/\tau_e)\,\delta_s + (K_e/\tau_e)\,\delta_t \tag{3.9}$$

The throttle servo is modelled as a second order plant in Equation 3.10.

$$\ddot{\delta}_t = -2\zeta\omega\dot{\delta}_t - \omega^2\delta_t + \omega^2 u_t \tag{3.10}$$

The signal flow graph for this system is shown in Figure 3.23. The following state space equations are used to design the controller for altitude rate :

$$x = \begin{bmatrix} \dot{h} \\ \delta_s \\ \delta_t \\ \dot{\delta}_t \end{bmatrix} \tag{3.11}$$

$$\dot{x} = \begin{bmatrix} 0 & 0.0865 & 0 & 0 \\ 0 & -2 & 1675.5 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -157.91 & -17.77 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 157.91 \end{bmatrix} u_t \tag{3.12}$$

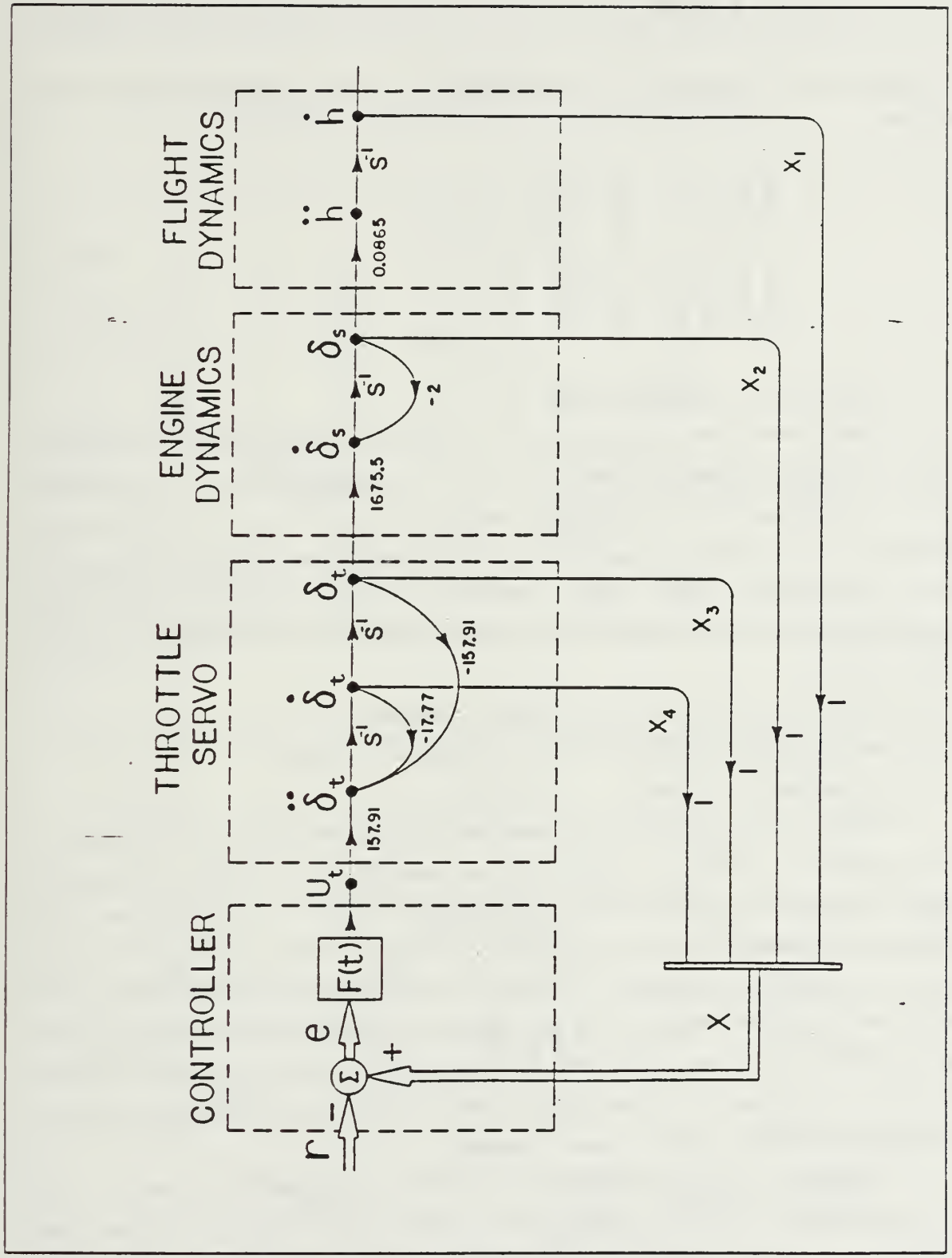

Figure 3.23 Signal Flow Diagram for Altitude Rate Control

$$u_t = \mathbf{F}\{x - r\} \tag{3.13}$$

If a unit step is commanded for the altitude rate, then the command input vector becomes

$$r = \begin{bmatrix} \dot{h}_c \\ \delta_{sc} \\ \delta_{tc} \\ \delta_{tc} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.14}$$

2. **Altitude Rate Controller Design**

The system given in Equations 3.12 and 3.14 is entered into the OPTCON program and a controller is designed according to a procedure similar to that explained for the roll rate controller. As before, a sampling rate of 20 Hz is used for all runs. Only two runs are hereafter presented because the lessons learned during the design of the roll rate controller apply equally as well to the design procedure for the altitude rate controller. The performance specifications for this control system are designated to be as follows :

1. Zero steady state error for a step input is required.
2. The two percent settling time, $t_{2\%}$, is less than 5 seconds.
3. No overshoot is allowed.

The first run is made using a baseline cost function. The results obtained for this run appear in Table 11. Notice that an incredibly large number of stages are required in order for $\mathbf{F}_{ss}$ to be achieved using this cost function. If the gains were to be implemented dynamically at 20 Hz, more than 38 seconds would be required before the steady state gains are available. This clearly is not desirable since the settling time must be less than five seconds. The unit step time response using steady state gains from this first run is shown in Figure 3.24. Even after 20 seconds, the desired altitude rate is not yet achieved. The cost function used to generate this solution is deemed to be unsatisfactory and a better solution is sought.

The final run for the altitude rate controller is summarized in Table 12. The cost function for this run places 100 times more emphasis on errors in the x_1 state than on errors in the x_2 and x_3 states. The altitude rate time response shown in Figure 3.25 exhibits acceptable performance characteristics. By choosing the cost function wisely, it becomes possible to design a satisfactory controller for this system.

TABLE 11

INITIAL ALTITUDE RATE PARAMETERS

Cost Function

$$\hat{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = 1$$

Steady State Gains				Number of Stages Required	Percent Overshoot	Settling Time (sec)
f_1	f_2	f_3	f_4			
-0.0544	-0.0461	-6.2776	-0.1948	763	0.00	> 20.0

TABLE 12

FINAL ALTITUDE RATE PARAMETERS

Cost Function

$$H = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 \\ 0 & 0 & .01 & 0 \\ 0 & 0 & 0 & .01 \end{bmatrix} \qquad R = 1$$

Steady State Gains				Number of Stages Required	Percent Overshoot	Settling Time (sec)
f_1	f_2	f_3	f_4			
-0.3485	-0.0312	-4.5999	-0.1569	128	0.00	4.65

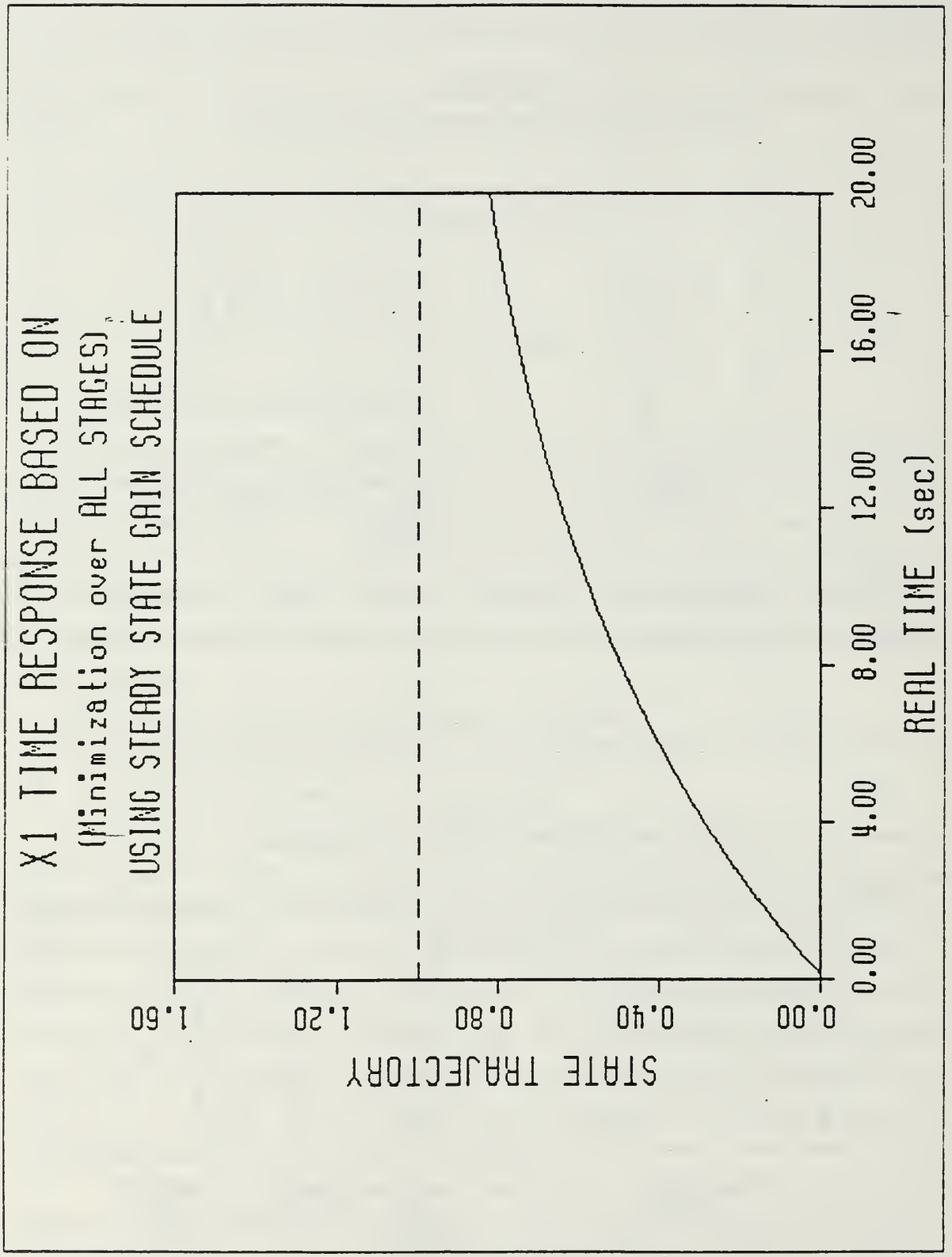

Figure 3.24 Initial Altitude Rate Time Response

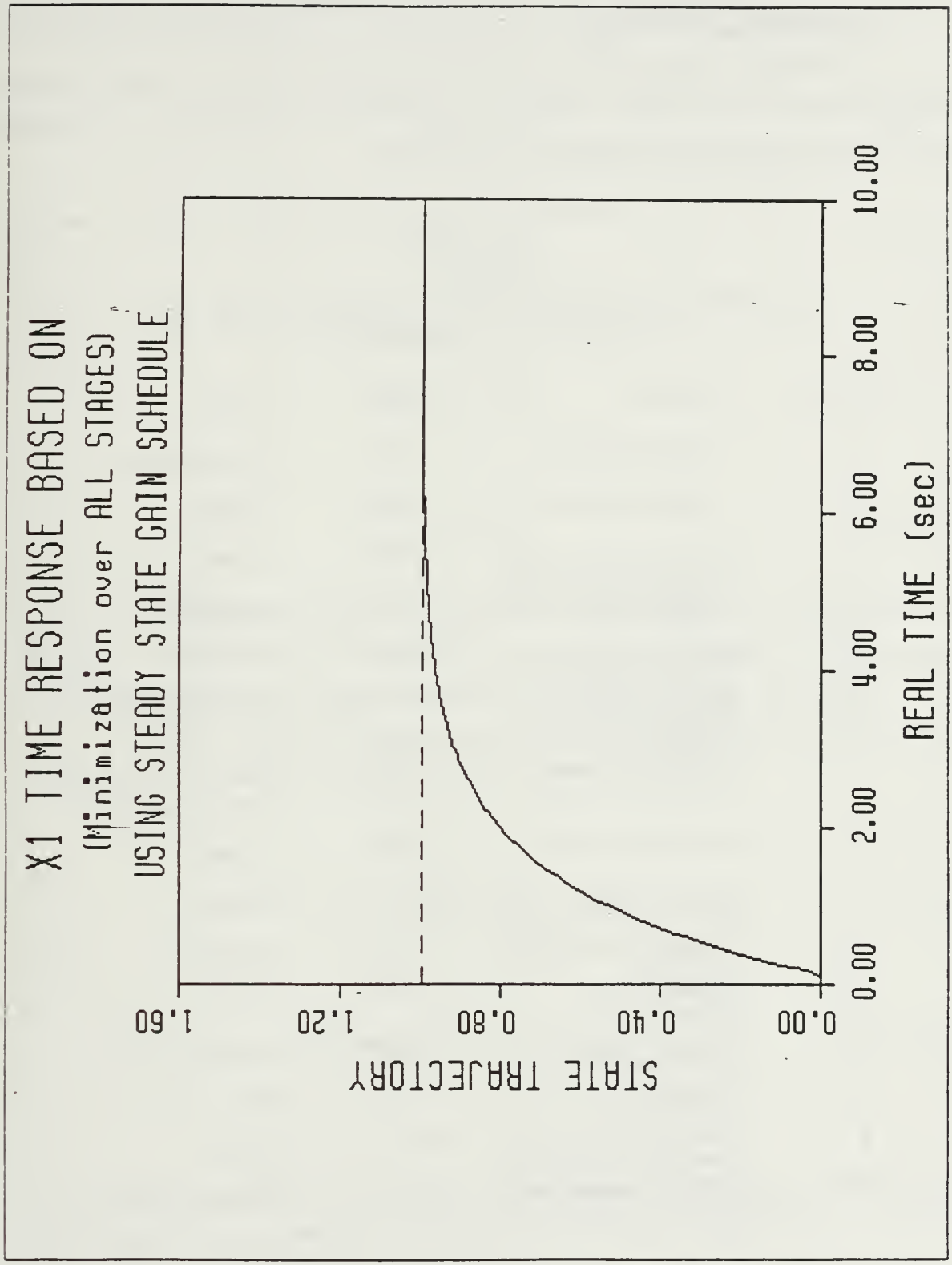

Figure 3.25 Final Altitude Rate Time Response

D. AROD PITCH ANGLE AND YAW ANGLE CONTROLLER

1. The Pitch and Yaw System

Gyroscopic coupling between the pitch and yaw dynamics of AROD creates an interesting control problem. Refer to Table 13 for an explanation of the terms involved in the pitch and yaw equations which follow.

TABLE 13
VARIABLE DEFINITIONS FOR AROD PITCH AND YAW EQUATIONS OF MOTION

Variable	Definition	Value	Units
q	Vehicle Pitch Rate	TBD	radians/second
θ	Vehicle Pitch Angle	TBD	radians
C_q	Pitch to Yaw Gyroscopic Coupling	-6.78	seconds^{-1}
M_e	Elevator Effectiveness Coefficient	-14.51	seconds^{-2}
δ_e	Elevator Servo Deflection Angle	$\leq 30°$	radians
$\dot{\delta}_e$	Elevator Servo Deflection Velocity	$\leq 50°$/sec	radians/second
u_e	Control Input to Elevator Servo	TBD	volts
r	Vehicle Yaw Rate	TBD	radians/second
ψ	Vehicle Yaw Angle	TBD	radians
C_r	Yaw to Pitch Gyroscopic Coupling	6.75	seconds^{-1}
N_r	Rudder Effectiveness Coefficient	-16.68	seconds^{-2}
δ_r	Rudder Servo Deflection Angle	$\leq 30°$	radians
$\dot{\delta}_r$	Rudder Servo Deflection Velocity	$\leq 50°$/sec	radians/second
u_r	Control Input to Rudder Servo	TBD	volts
ζ	Elevator/Rudder Servo Damping Coefficient	0.707	unitless
ω	Elevator/Rudder Servo Natural Frequency	12.57	radians/second

The pitch and yaw equations of motion are given in Equations 3.15 through 3.18.

$$q = C_r r + M_e \delta_e \tag{3.15}$$

$$\theta = \int q \, dt \tag{3.16}$$

$$r = C_q q + N_r \delta_r \tag{3.17}$$

$$\psi = \int r \, dt \tag{3.18}$$

Note that crosscoupling between the pitch and yaw equations enters via the two gyroscopic coupling terms, C_q and C_r. The values listed in Table 13 for these two coefficients are based on an assumed constant propeller velocity of 7200 RPM.

As before, the elevator and rudder control vane servos are modelled as second order systems according to Equations 3.19 and 3.20.

$$\ddot{\delta}_e = -2\zeta\omega\dot{\delta}_e - \omega^2\delta_e + \omega^2 u_e \tag{3.19}$$

$$\ddot{\delta}_r = -2\zeta\omega\dot{\delta}_r - \omega^2\delta_r + \omega^2 u_r \tag{3.20}$$

A coupled eighth order system results from Equations 3.15 through 3.20. The signal flow diagram which represents this MIMO system is given in Figure 3.26.

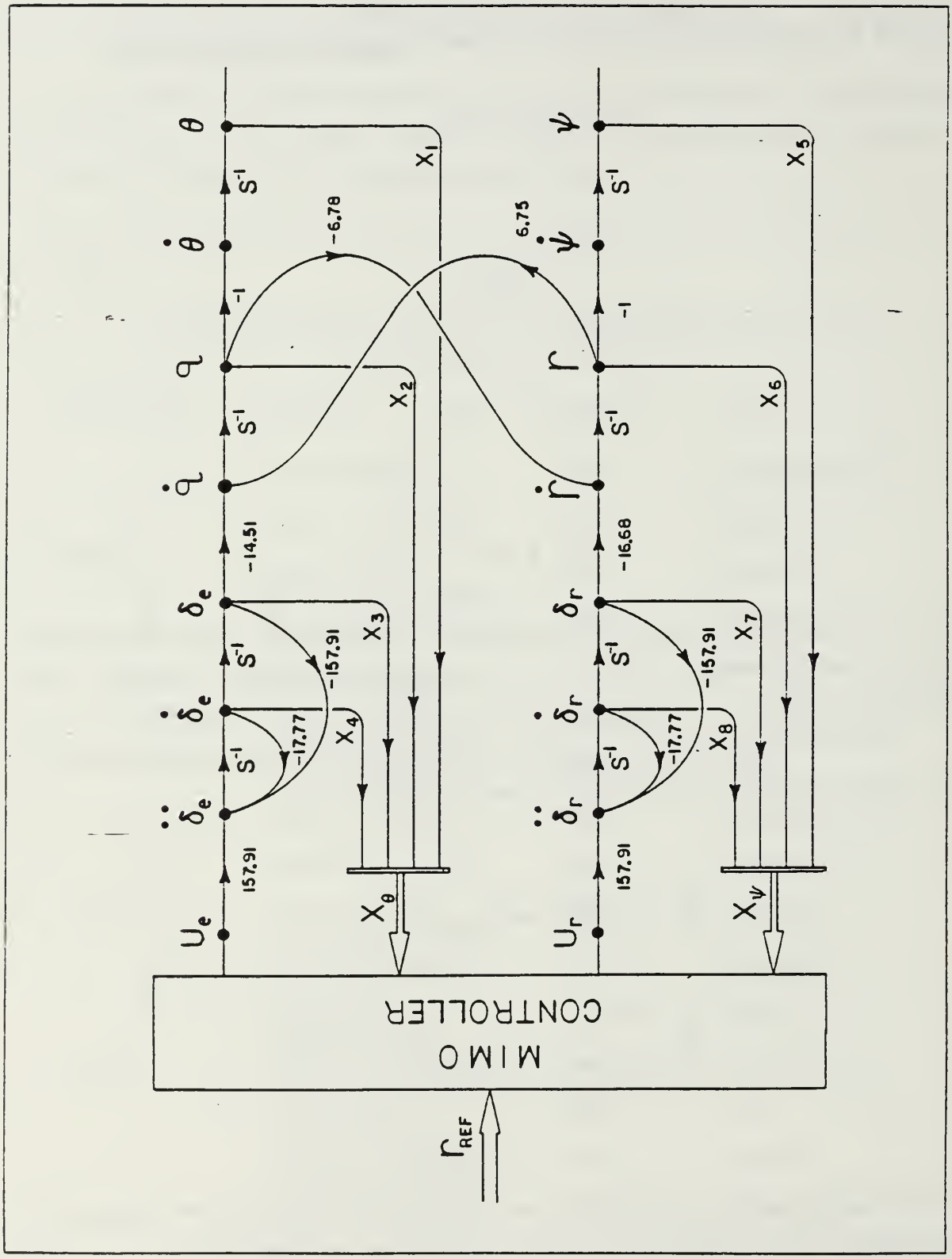

Figure 3.26 Signal Flow Diagram for Pitch and Yaw Angle Control

Defining the eight states to be :

$$x = \left[\theta \;\; q \;\; \delta_e \;\; \dot{\delta}_e \;\; \psi \;\; r \;\; \delta_r \;\; \dot{\delta}_r\right]^t \tag{3.21}$$

the state space equation for the pitch and yaw coupled system is defined as

$$\dot{x} = Ax + Bu \tag{3.22}$$

where

$$A = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -14.51 & 0 & 0 & 6.75 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -157.91 & -17.77 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -6.78 & 0 & 0 & 0 & 0 & -16.68 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -157.91 & -17.77 \end{bmatrix} \tag{3.23}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 157.91 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 157.91 \end{bmatrix} \tag{3.24}$$

and the multi-input control vector is

$$u = \begin{bmatrix} u_e \\ u_r \end{bmatrix} = F_{mimo}\{x - r\} \tag{3.25}$$

2. **Pitch Angle and Yaw Angle Controller Design**

a. Methodology

Notice in Equation 3.25 that the control input, **u**, is a (2 × 1) vector. Up to this point in the AROD control design, the control input has been limited to a scalar signal. The multi-input control that results from gyroscopic coupling between pitch and yaw requires special attention. Consider the following points :

1. The optimal feedback gain matrix, **F**, is determined from the solution of the discrete matrix Riccati equation. This solution requires that the inverse of the term $\{\mathbf{\Gamma}^t \mathbf{P}(K-1) \mathbf{\Gamma} + \mathbf{R}\}$ in Equation 2.28 be determined.
2. The cost function for a SISO system requires that the control weighting factor, R, be a scalar.
3. The cost function for an n^{th} order MIMO system with ℓ control inputs requires that **R** be an ($\ell \times \ell$) matrix.

Thus, for a SISO system, the solution for **F** is greatly simplified because the term in Equation 2.28 is a scalar quantity. For a MIMO system, however, a matrix inversion routine is required in order to solve for the optimal gains. Although computationally possible, it is decided for the purpose of this work that no matrix inversion routine is to be included in the current version of OPTCON. This implies that the ability of the OPTCON program to solve for optimal feedback gains is necessarily limited to SISO systems. This limitation is reasonable since a multitude of control problems can be reduced to single input systems. In the case of AROD, however, gyroscopic coupling is a permanent feature of the pitch and yaw dynamics. Thus, a MIMO system is inevitable. The four step tactic used to design a control system for this non-trivial problem is as follows :

1. First assume that the gyroscopic coupling terms, C_q and C_r, are zero so that the coupled eighth order system reduces to two independent fourth order systems.
2. Use OPTCON to solve for the optimal feedback gains for the two independent systems.
3. Implement the steady state gains so obtained for the fourth order uncoupled systems into a simulation model for the eighth order coupled system.
4. Experiment with various combinations and modifications to the (2 × 8) feedback matrix, $\mathbf{F}_{mimo}$, until a satisfactory time response is obtained for the pitch angle and yaw angle of the coupled system.

Note that the design procedure listed above does *not* represent the most direct method to design a MIMO controller using optimal control theory. Rather, this method is an attempt to solve a complex problem using a tool that is designed to solve simpler

problems. For this reason, the results may not necessarily be expected to meet the same high standards required of the two previous control designs. The target performance specifications for the pitch and yaw control system are stated to be :

1. Zero steady state error for a step input is required.
2. The two percent settling time, $t_{2\%}$, is less than 2 seconds.
3. Less than 10% overshoot is allowed.

(1) *Decoupling the System Equations.* The decoupling procedure results in two fourth order systems. The uncoupled pitch angle state space equations are :

$$x_\theta = \begin{bmatrix} \theta \\ q \\ \delta_e \\ \dot{\delta}_e \end{bmatrix} \tag{3.26}$$

$$\dot{x}_\theta = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -14.51 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -157.91 & -17.77 \end{bmatrix} x_\theta + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 157.91 \end{bmatrix} u_e \tag{3.27}$$

$$u_e = F_e \{x_\theta - r_\theta\} \tag{3.28}$$

If a unit step is commanded for the pitch angle, then the pitch command input vector becomes

$$r_\theta = \begin{bmatrix} \theta_c \\ q_c \\ \delta_{ec} \\ \dot{\delta}_{ec} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

The uncoupled yaw angle state space equations are :

$$x_\psi = \begin{bmatrix} \psi \\ r \\ \delta_r \\ \dot{\delta}_r \end{bmatrix} \tag{3.30}$$

$$\dot{x}_\psi = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -16.68 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -157.91 & -17.77 \end{bmatrix} x_\psi + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 157.91 \end{bmatrix} u_r \tag{3.31}$$

$$u_r = F_r \{x_\psi - r_\psi\} \tag{3.32}$$

If a unit step is commanded for the yaw angle, then the yaw command input vector becomes

$$r_\psi = \begin{bmatrix} \psi_c \\ r_c \\ \delta_{rc} \\ \dot{\delta}_{rc} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.33}$$

The similarity between the dynamics of the uncoupled pitch and yaw systems is advantageous. For example, it is found that the elevator and rudder control gains, F_e and F_r, which are generated by OPTCON have identical steady state values. For this reason, only *one* set of steady state gains, F_{ss}, needs to be generated for each cost function. The individual elements of F_{ss} are hereafter referred to as f_1, f_2, f_3, and f_4.

(2) *Solving for the Uncoupled Controller.* The solution for F_{ss} for the two fourth order systems is straightforward and follows the procedure established earlier in this chapter. Table 14 summarizes the data from the initial run. A unit step time response for the pitch or yaw angle controller implementing steady state gains from Table 14 is shown in Figure 3.27. Table 15 contains the data for the final run. The time response for this last controller appears in Figure 3.28. The steady state gains listed in Table 15 serve as the foundation upon which the coupled controller is subsequently designed.

TABLE 14

INITIAL UNCOUPLED PITCH OR YAW PARAMETERS

Cost Function

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = 1$$

Steady State Gains				Number of Stages Required	Percent Overshoot	Settling Time (sec)
f_1	f_2	f_3	f_4			
-0.1704	0.2417	-0.2490	-0.0858	96	0.00	4.15

TABLE 15

FINAL UNCOUPLED PITCH OR YAW PARAMETERS

Cost Function

$$H = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 \\ 0 & 0 & .01 & 0 \\ 0 & 0 & 0 & .01 \end{bmatrix} \qquad R = 5.0$$

Steady State Gains				Number of Stages Required	Percent Overshoot	Settling Time (sec)
f_1	f_2	f_3	f_4			
-0.3961	0.2808	-0.4158	-0.0259	58	3.98	2.50

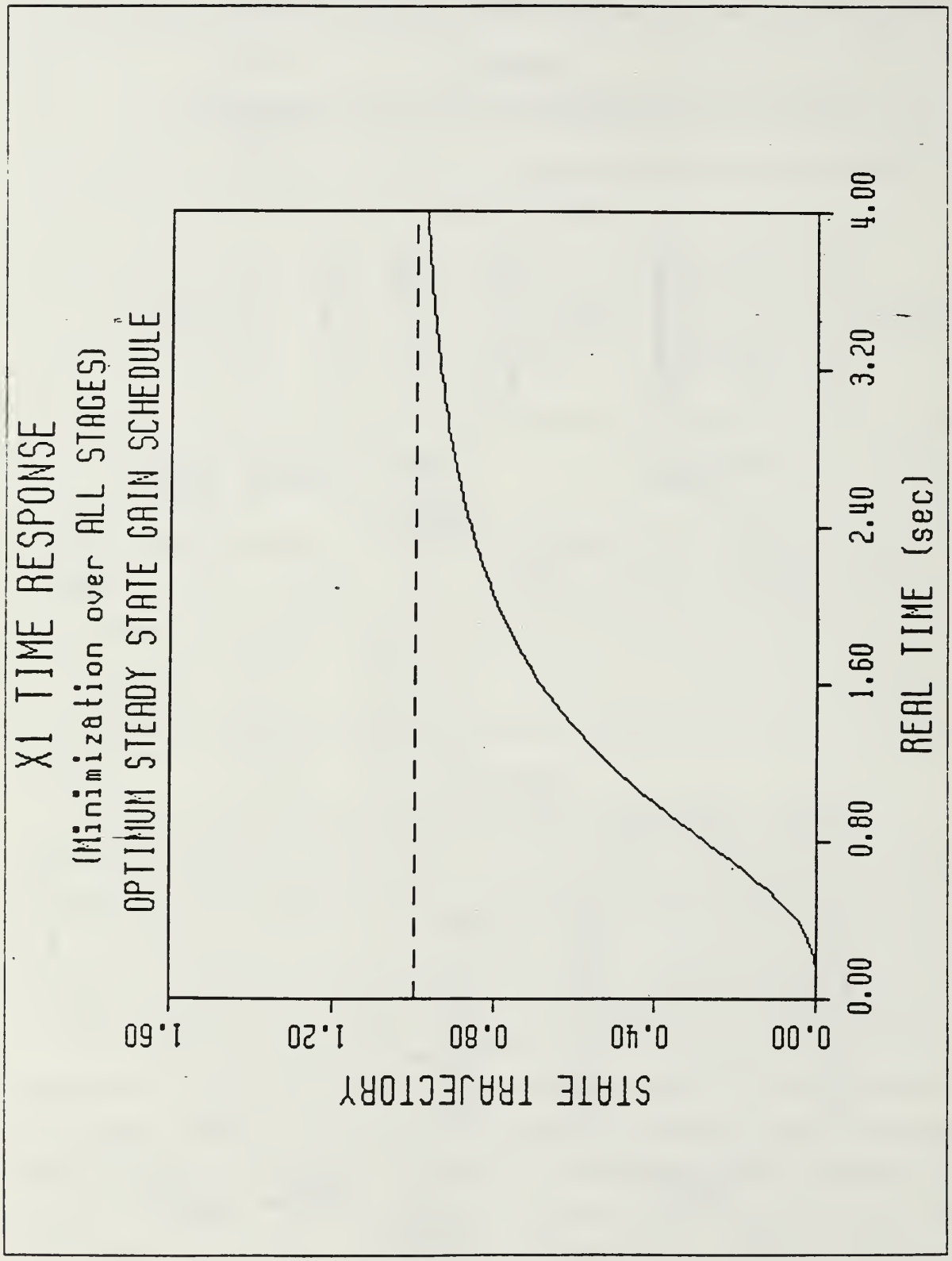

Figure 3.27 Initial Uncoupled Pitch or Yaw Angle Time Response

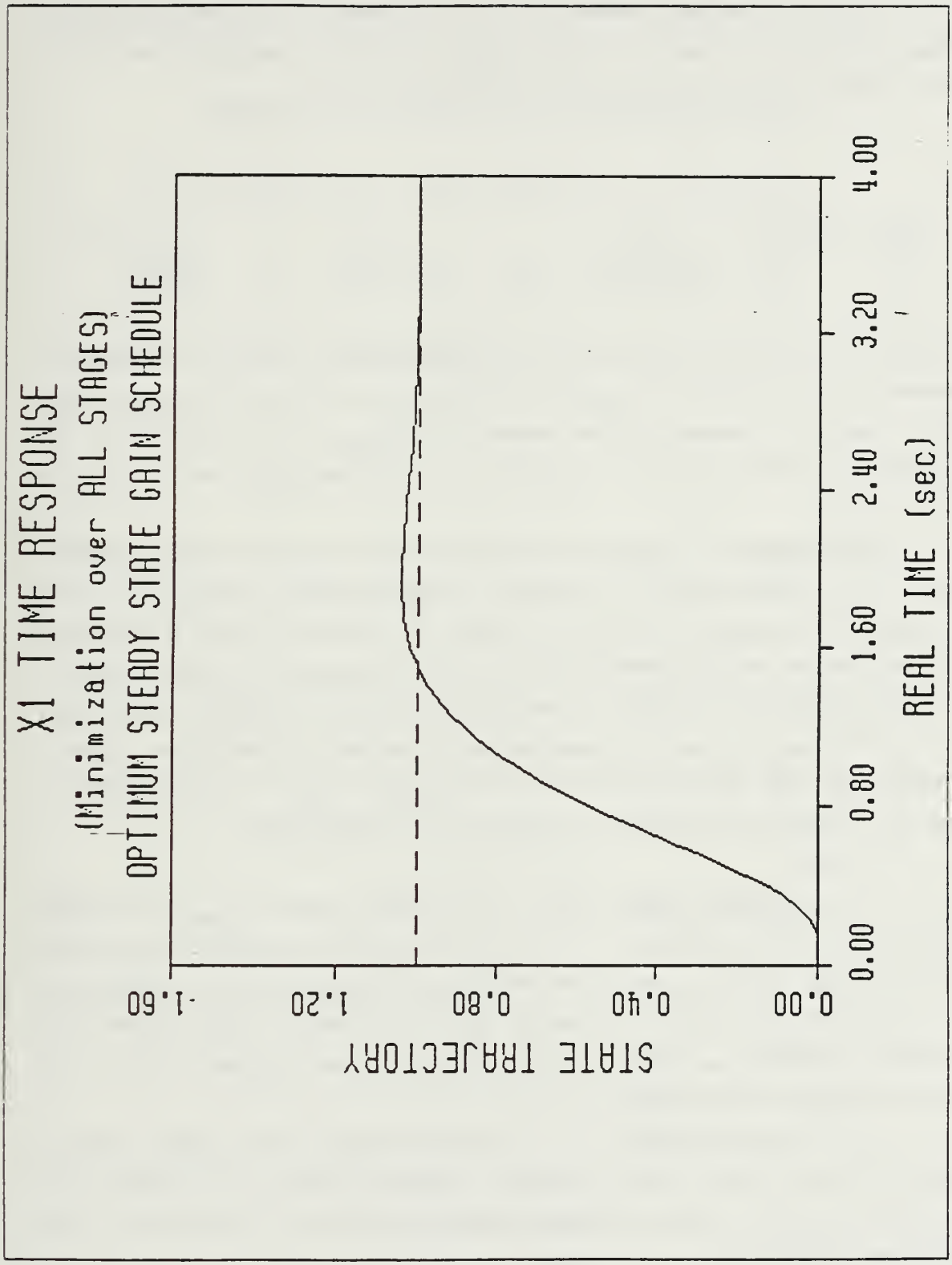

Figure 3.28 Final Uncoupled Pitch or Yaw Angle Time Response

(3) *The Coupled Feedback Matrix.* In order to implement the SISO feedback gain vectors, $\mathbf{F}_e$ and $\mathbf{F}_r$, into the coupled MIMO state equations, the (2 × 8) feedback matrix, $\mathbf{F}_{mimo}$, is formed as shown in Equation 3.34.

$$\mathbf{F}_{mimo} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & \square_{15} & \square_{16} & \square_{17} & \square_{18} \\ \square_{21} & \square_{22} & \square_{23} & \square_{24} & f_1 & f_2 & f_3 & f_4 \end{bmatrix} \quad (3.34)$$

The $\square_{ij}$ elements of $\mathbf{F}_{mimo}$ represent those feedback elements which are not specifically generated by OPTCON. The success of the final control system is contingent upon proper selection of values for these elements of $\mathbf{F}_{mimo}$. For the purpose of the following discussion, the reader is encouraged to refer to the signal flow graph shown in Figure 3.26.

(4) *Analysis.* There are numerous ways to select the eight unspecified feedback gains in Equation 3.34. The seven schemes examined during the course of this design are summarized in Table 16. The first two columns in Table 16 identify the controller structure used to generate the elevator and rudder control signals, u_e and u_r. The third column refers the reader to the appropriate figure containing the pitch or yaw angle time response for that particular set of parameters. The last two columns summarize the time response data for each controller design. At the bottom of Table 16 are listed the exact numerical values for the controller gains.

b. Results

(1) *Controller Number One.* The feedback matrix, $\mathbf{F}_{mimo}$, in this first design assumes that the four states of the yaw equations have *no* influence on the pitch control input, u_θ. Similarly, the four states of the pitch equation have *no* influence on the yaw control input, u_ψ. As expected, due to the known coupling that exists between the pitch and yaw dynamics of the vehicle, the time response in Figure 3.29 exhibits unsatisfactory performance.

(2) *Controller Number Two.* For this design, the pitch angle state, x_1, influences the yaw control input, u_ψ, while the yaw angle state, x_5, contributes to the pitch control input, u_θ. From the time response in Figure 3.30, it is apparent that this design is not satisfactory.

TABLE 16
PITCH/YAW CONTROLLER DESIGN SCHEMES

Design Number	F_{mimo} Organization	Time Response Figure	Percent Overshoot	Settling Time (sec)
1	$f_1\ f_2\ f_3\ f_4\ 0\ 0\ 0\ 0$ $0\ 0\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.29	34.3	18.9
2	$f_1\ f_2\ f_3\ f_4\ f_1\ 0\ 0\ 0$ $f_1\ 0\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.30	45.1	N/A
3	$f_1\ f_2\ f_3\ f_4\ f_1\ f_2\ 0\ 0$ $f_1\ f_2\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.31	N/A	N/A
4	$f_1\ f_2\ f_3\ f_4\ f_1\ f_2\ 0\ 0$ $f_1\ -f_2\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.32	2.15	1.85
5	$f_1\ f_2\ f_3\ f_4\ f_1\ f_2\ 0\ 0$ $f_1\ -f_*\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.33	0.00	1.88
6	$f_1\ f_2\ f_3\ f_4\ f_1\ f_2\ 0\ 0$ $f_1\ -f_2\ f_3\ f_4\ f_1\ f_2\ f_3\ f_4$	3.34	0.00	N/A
7	$f_1\ f_2\ f_3\ f_4\ 0\ f_2\ 0\ 0$ $0\ -f_2\ 0\ 0\ f_1\ f_2\ f_3\ f_4$	3.35	22.5	8.57

$f_1 = -0.3961$
$f_2 = 0.2808$
$f_* = 0.2954$
$f_3 = -0.4158$
$f_4 = -0.0259$

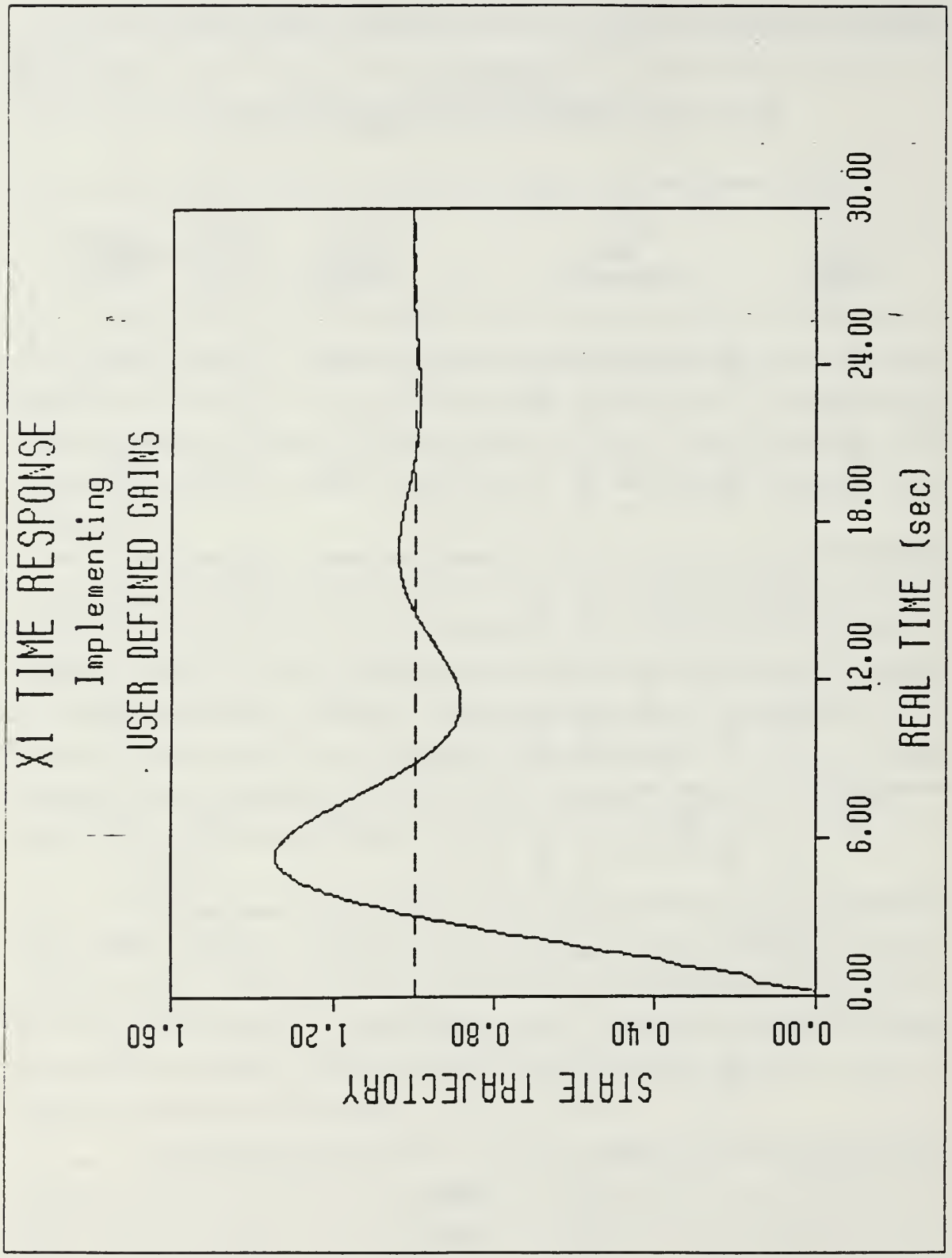

Figure 3.29 Pitch / Yaw Angle Time Response for Controller Number 1

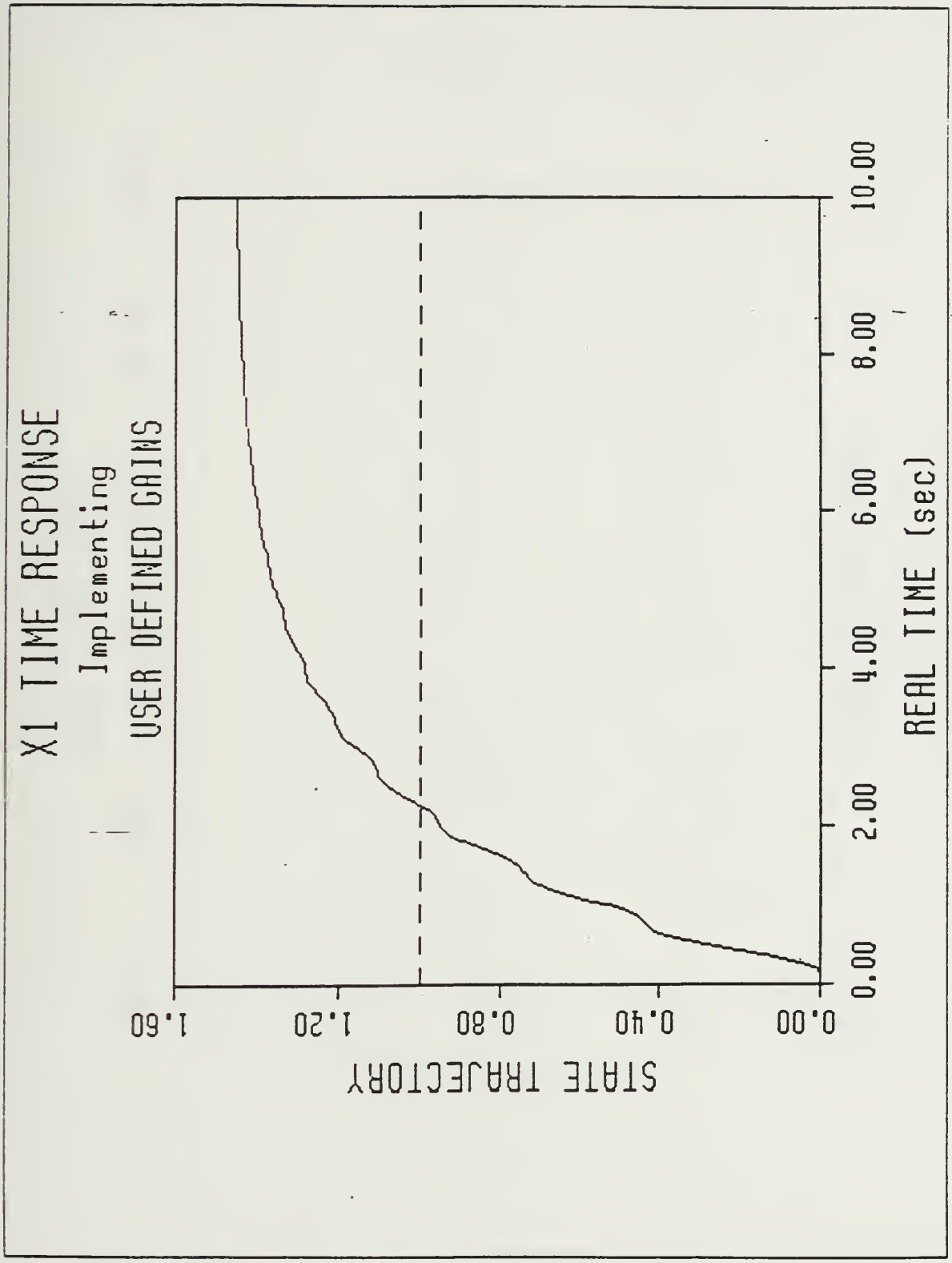

Figure 3.30 Pitch / Yaw Angle Time Response for Controller Number 2

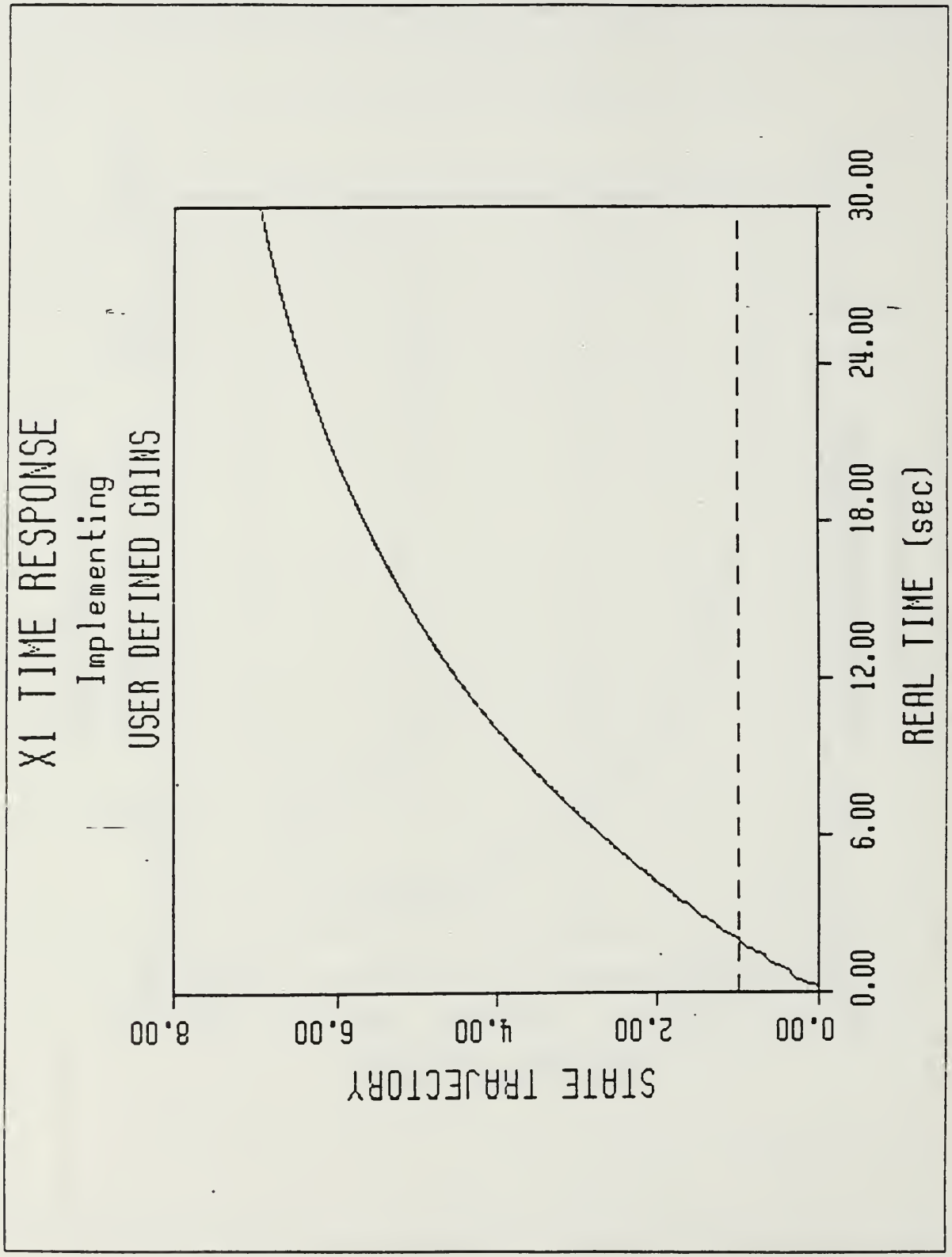

Figure 3.31 Pitch / Yaw Angle Time Response for Controller Number 3

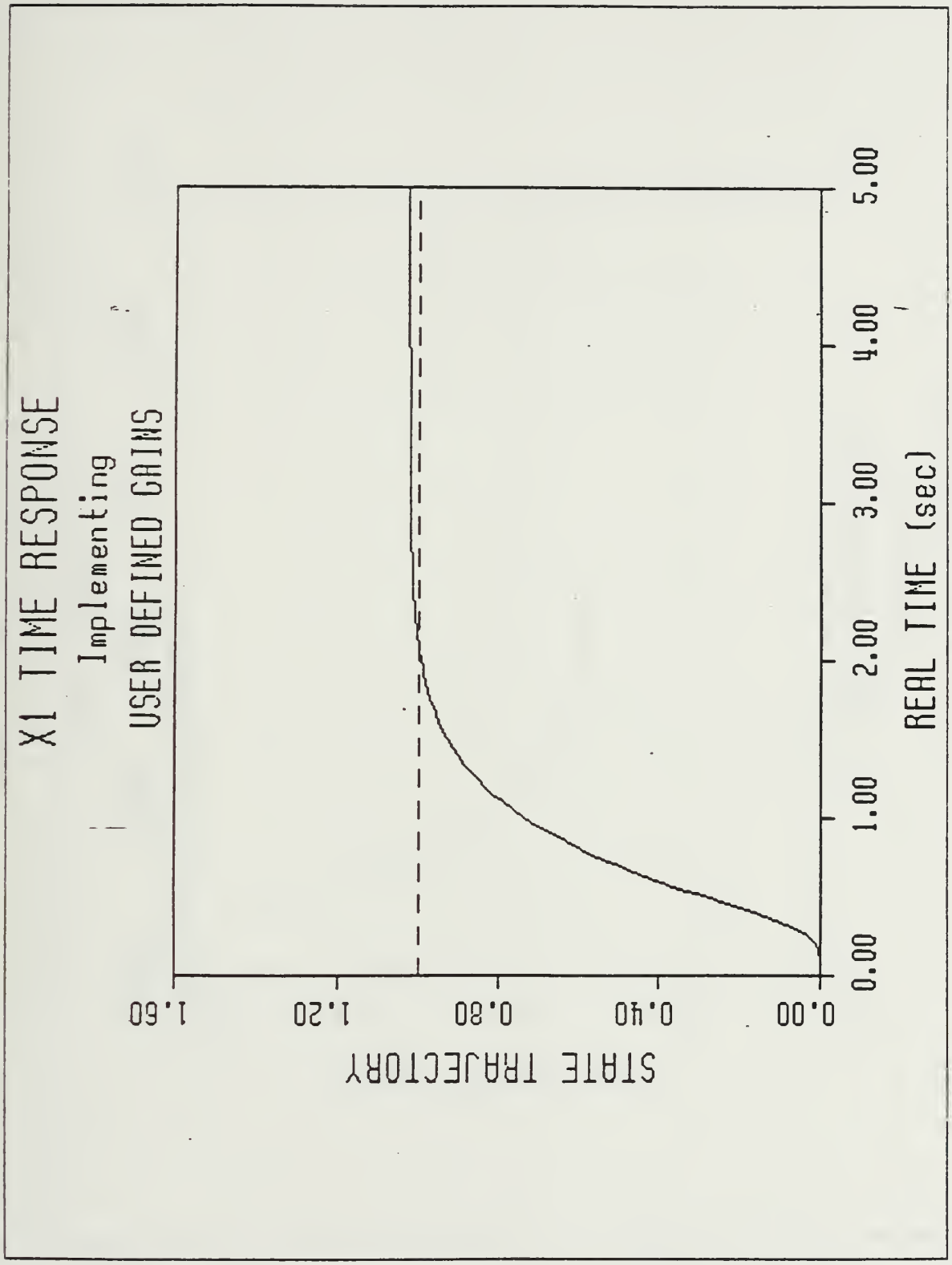

Figure 3.32 Pitch / Yaw Angle Time Response for Controller Number 4

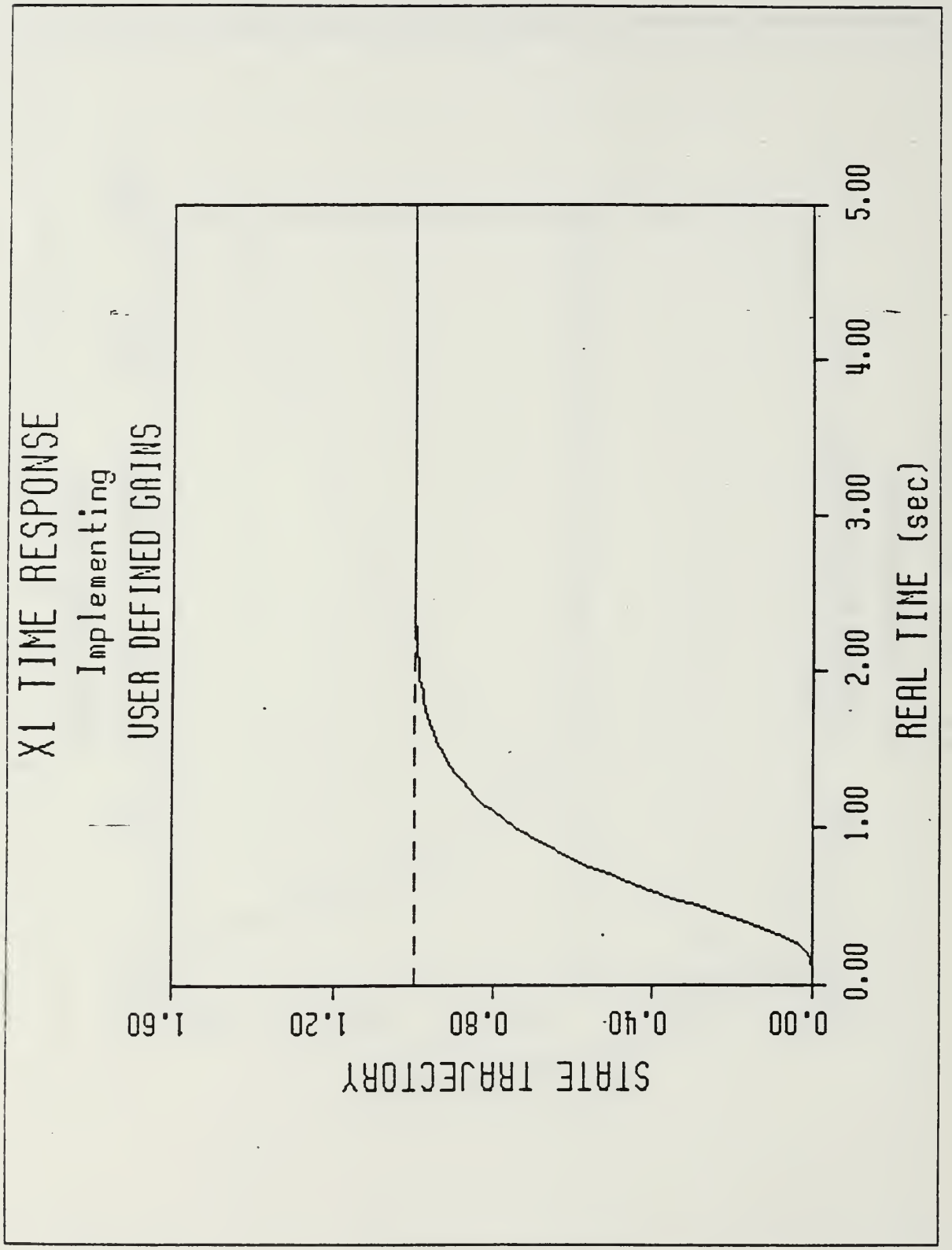

Figure 3.33 Pitch / Yaw Angle Time Response for Controller Number 5

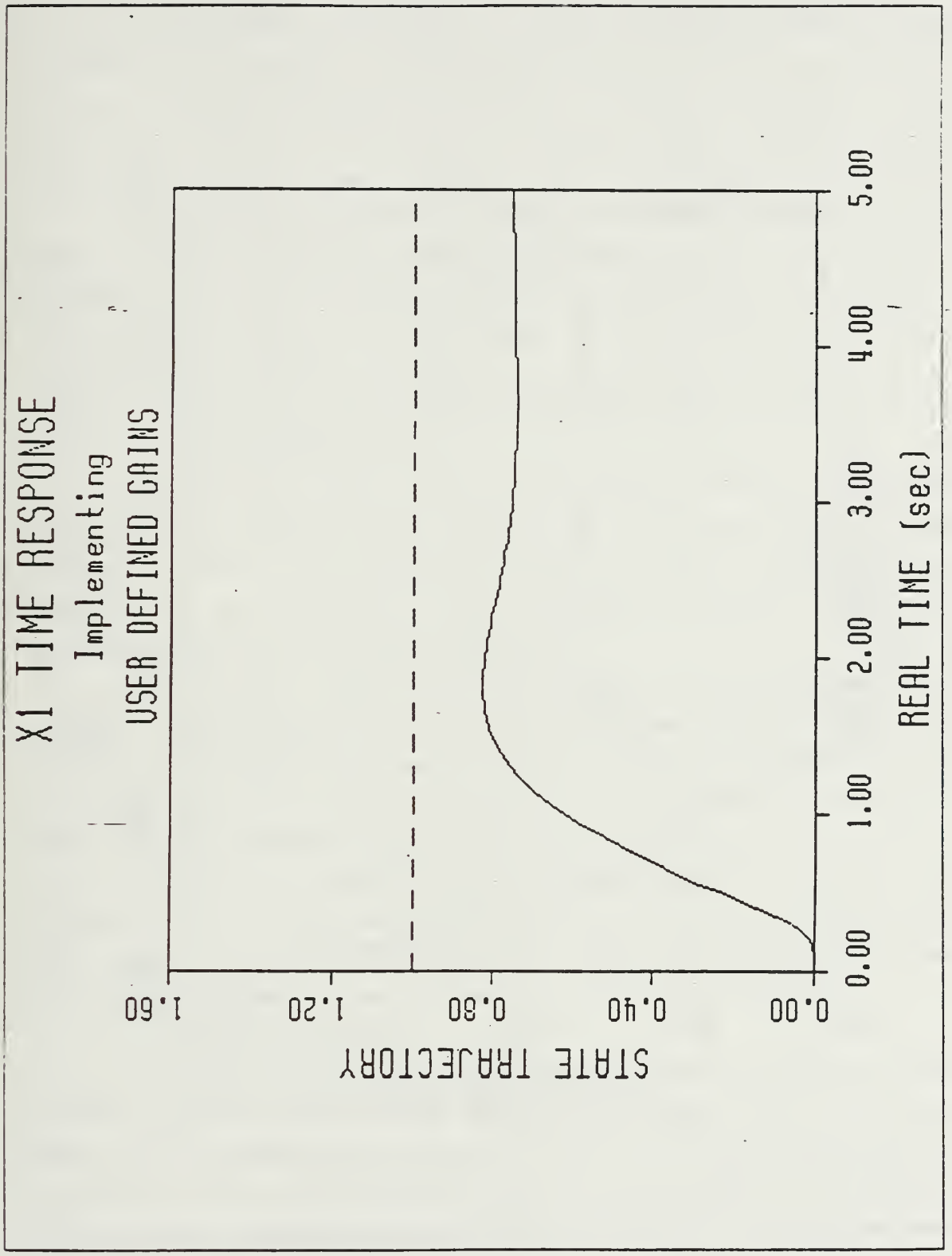

Figure 3.34 Pitch / Yaw Angle Time Response for Controller Number 6

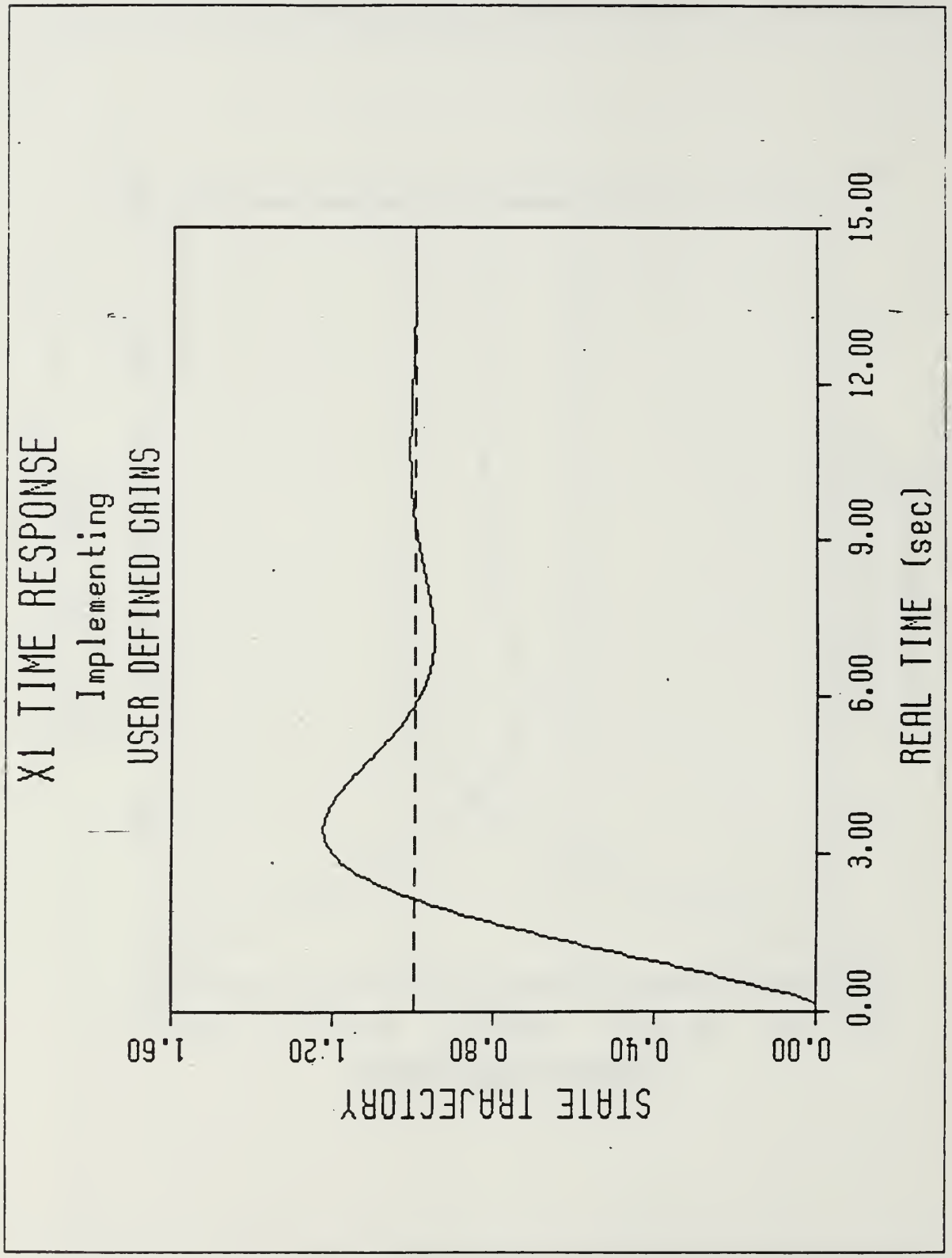

Figure 3.35 Pitch / Yaw Angle Time Response for Controller Number 7

(3) *Controller Number Three.* Because the gyroscopic coupling between the pitch and yaw equations is directly related to the pitch *rate*, x_2, and the yaw *rate*, x_6, it is decided to include these two states in the makeup of the yaw control and pitch control respectively. This seems like a logical design tactic at first but the resulting time response in Figure 3.31 proves otherwise. This design is clearly unstable.

(4) *Controller Number Four.* Notice in the coupled system signal flow diagram in Figure 3.26 that the coupling coefficients, C_q and C_r, are nearly equal in magnitude but *opposite* in sign. This realization causes the designer to hypothesize that the sign of $\square_{22}$ in F_{mimo} should be reversed from the value previously used in controller design number 3. As shown in the time response of Figure 3.32, this technique yields promising results. Although a steady state error of 2.15% exists, there is merit in pursuing this design further.

(5) *Controller Number Five.* By finetuning the value substituted into $\square_{22}$ in F_{mimo}, the time response for pitch angle or yaw angle is made to satisfy the desired performance criteria. The time response in Figure 3.33 exhibits no steady state error, zero overshoot, and a settling time, $t_{2\%}$, of slightly less than two seconds. This controller design, then, is completely satisfactory.

(6) *Controller Number Six.* For this design, *all eight* states are allowed to influence both u_θ and u_ψ. The time response so obtained is shown in Figure 3.34. Even though the steady state angle is only 75% of the commanded value, this controller design appears to be potentially useful. By tuning the gains iteratively, it is believed that zero steady state error is achievable with this design.

(7) *Controller Number Seven.* This final design is a modification to controller number 3. In this case, only the pitch *rate* and yaw *rate* contribute to the crosscoupled control signals. This design effort results in unsatisfactory performance as shown in Figure 3.35.

c. The Final Design

Controller number 5 is selected as the best design for a pitch/yaw angle controller. The time response for this design appears in Figure 3.33. Note that this design is *based* on feedback gains generated by OPTCON but that a modification to f_2 is required in order to obtain the final design. Thus, the controller is not *optimal*, by formal definition, even though optimal control theory provides the foundation for its development.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The lessons learned during the course of this work are as follows :

1. The cost function weighting parameters, **H**, **Q**, and **R**, play vital roles in determining the magnitude of the steady state optimal feedback gain matrix, $\mathbf{F}_{ss}$. These control gains, in turn, significantly affect the time response of the controlled system.
2. There is no magic formula to determine proper values for the weighting factors. A reasonable starting point is to use the baseline cost function in which all diagonal elements of **H**, **Q**, and **R** are assigned the value of unity and all off-diagonal elements are zero.
3. The process of trial and error is prerequisite to the successful design of an optimal control system. Only through an iterative procedure does the engineer establish the true nature of the problem at hand.
4. There are obvious trends to be aware of. These include :
   a. The sampling frequency, f_s, must be fast enough to avoid aliasing effects. As predicted, the use of a sampling frequency that is five to ten times faster than the Nyquist frequency seems to be adequate.
   b. As the selected sampling frequency increases, the optimal gains generated also tend to increase in magnitude.
   c. The control weighting factor, R, for a SISO system can be used as a parameter to systematically alter the time response of the system. As the relative weight on the control effort increases, the steady state gains tend to decrease in magnitude. This generally produces a slow system that exhibits little or no overshoot. On the other hand, if the value of R is decreased, the steady state gains can become so large that a very fast and highly oscillatory system results.
5. The controller design for a MIMO system is significantly more involved than the design for a SISO system. If the engineer can logically and accurately decouple the MIMO system into multiple SISO systems, then the design effort becomes much easier. As shown in the pitch and yaw controller for AROD, this method *is* feasible.

B. RECOMMENDATIONS FOR FURTHER WORK

The following areas present valid opportunities for useful expansion of this work :

1. A parameter identification procedure which aids the design engineer in determining or estimating the **A**, **B**, $\mathbf{\Phi}$, and $\mathbf{\Gamma}$ plant matrices is needed. The use

of a Fast Fourier Transform (FFT) algorithm is one possible solution to this requirement. Such a program could be used in conjunction with, but not necessarily integrated into, the existing OPTCON package.

2. The OPTCON program is limited in that it does not generate optimal feedback gains for a MIMO system. A matrix inversion routine is needed so that the discrete matrix Riccati solution can be determined for any ($n \times \ell$) $\mathbf{B}$ or $\mathbf{\Phi}$ matrix.

3. The theory of optimal control assumes availability of *all* states for feedback. The design process must account for the fact that all states are not always measured. In the case of AROD, the servo position and rate states are not available for feedback. This means that an observer must be designed in order to provide the missing state information.

4. The three control systems which are herein designed must be evaluated on the actual vehicle. Although computer simulations provide a wealth of insight, the proof of a good design rests in the ability of the system to function in the outside world.

APPENDIX A
THE OPTCON PROGRAM

1. OVERVIEW

The purpose of this appendix is to describe in detail the OPTCON computer program which was developed in support of this thesis. OPTCON derives its name from OPTimal CONtrol. A previous edition of OPTCON by Professor H.A. Titus of the Naval Postgraduate School provided the starting point for the work that follows. The original OPTCON program allowed the user to input a state space system either in the continuous time domain or in the discrete time domain. Using matrix calculations to solve Equations 2.28, 2.29, and 2.30, this first version of OPTCON generated a table of feedback gains and immediately terminated execution. The motivation for improving the original OPTCON is fourfold.

1. The design process is an iterative technique. The OPTCON program needs to be flexible enough to allow minor changes to be made to *specific* parameters without the requirement to re-initialize *all* of the cost function and system values.
2. The gain trajectory table is not a convenient means by which to analyze the solution to an optimal control problem. A graph of the feedback gains verses time provides better insight.
3. A time response of the state space is needed in order to allow the designer to quickly evaluate the performance of the system.
4. The program should be user friendly. The original OPTCON demanded that the user flawlessly enter the correct response to *every* question on the *first* attempt. Woe be it to the user who accidentally types a letter in response to a question that requires a numerical answer. The program aborts and any effort that was spent in entering information is wasted. The frustration factor for such an unfriendly program is likely to leave the program sitting on the shelf with nobody to use it.

With these points in mind, the OPTCON program is rewritten to provide an interactive, menu driven, user-friendly, optimal control design tool that capitalizes on the graphical capabilities of modern microcomputers. The program is written in MICROSOFT Fortran and is listed in Appendices B, C, and D. Appendix B contains the driver program, MAIN. Appendix C contains the majority of the subroutines which are called by MAIN. Appendix D contains the plotting subroutine, GRAPH, which makes use of the PLOT88 graphics software package.

In order to use OPTCON to its full potential, the user needs access to the following :

1. A microcomputer capable of executing MICROSOFT Fortran based programs. During the development of OPTCON, an IBM AT computer configured with 640 kbyte RAM and Intel's 80287 math co-processor was used.
2. Fortran, PLOT88, and Math libraries.
3. A monochrome or color graphics monitor.
4. An Epson or LaserJet printer.

Figure A.1 is provided to give the user a broad overview of the basic program flow of OPTCON. The blocks outlined by solid lines represent program segments that *must* be performed during the initial execution of OPTCON. The blocks outlined by dashed lines represent program segments that are *optional.* The numbers that appear to the left of each block are referred to during in Section 2.d of this appendix.

The remainder of this appendix illustrates the solution to a simple example problem using the OPTCON program. The intent here is not to focus on the specific example problem or on its solution but, rather, to focus on the capabilities and use of OPTCON.

2. AN EXAMPLE PROBLEM

a. The Second Order Integrator

Consider the continuous time system shown in Figure A.2. The state space equations for this second order plant are derived by defining the output of each integrator to be a state. Using Equations 2.7 through 2.10 as a basis, the state equations become :

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}(t) \qquad \text{(A.1)}$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}(t) \qquad \text{(A.2)}$$

$$u(t) = f_1(x_1 - r_1) + f_2(x_2 - r_2) \qquad \text{(A.3)}$$

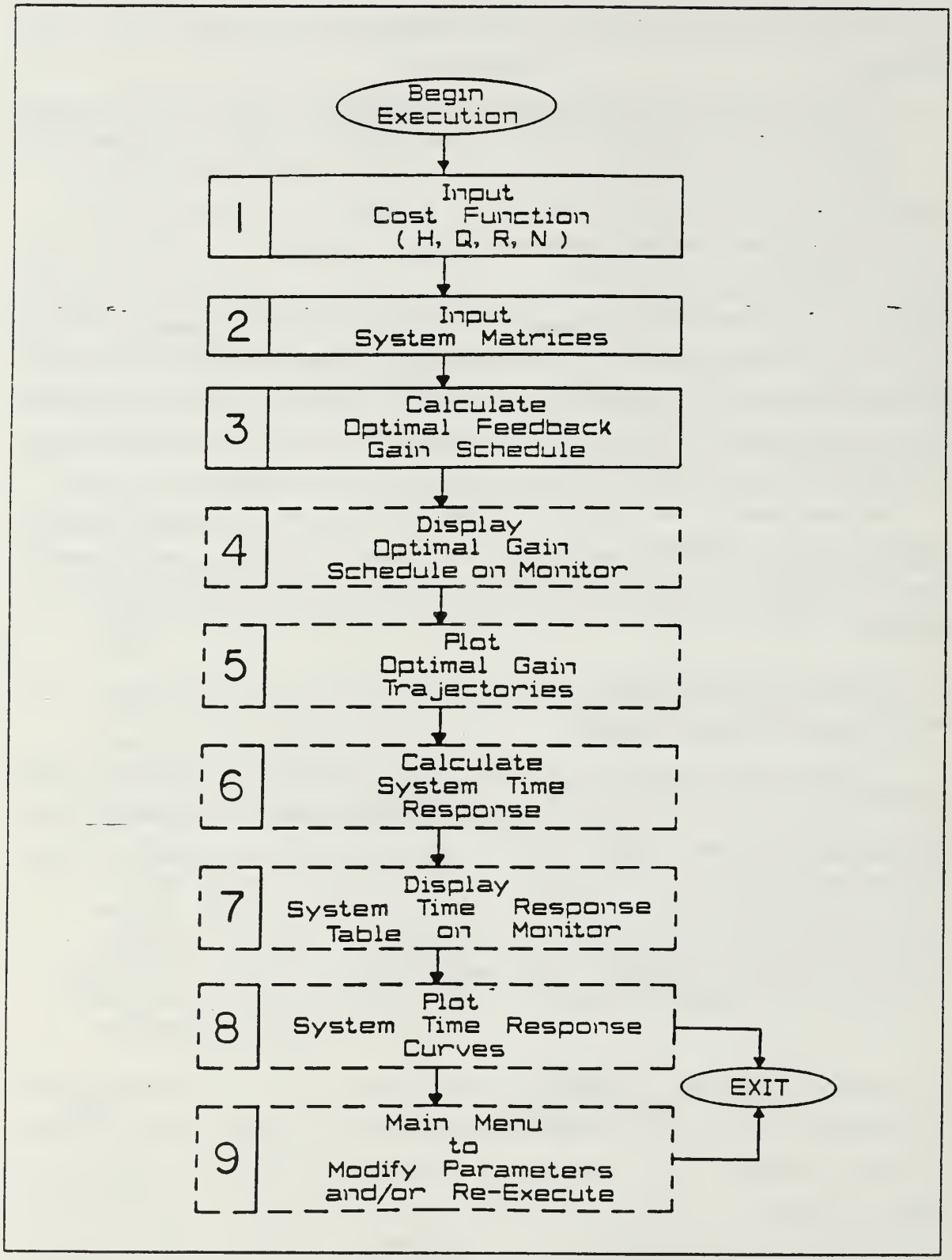

Figure A.1 OPTCON Program Flow Diagram

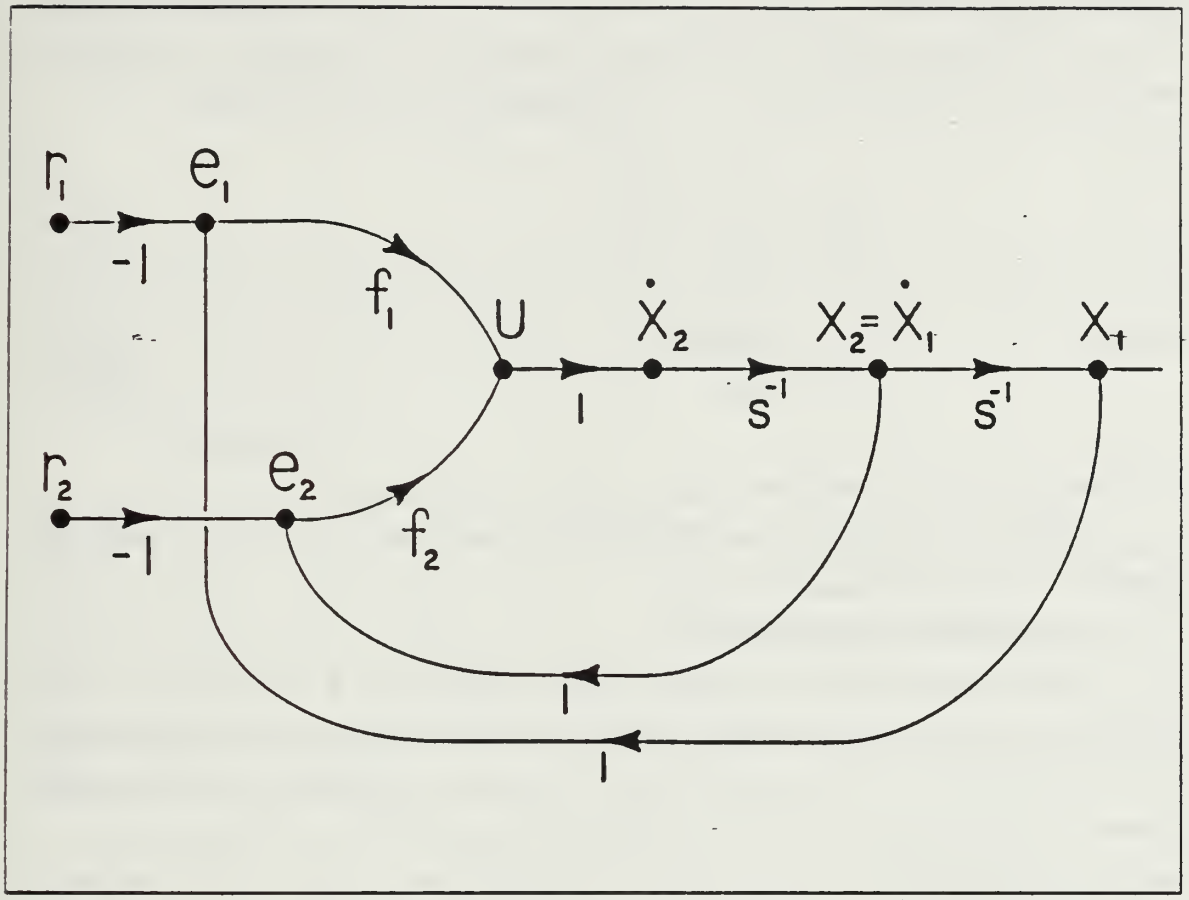

Figure A.2 Second Order Integrator Signal Flow Diagram

If the system is sampled every T seconds, Equation 2.20 yields :

$$\Pi = \begin{bmatrix} T & T^2/2 \\ 0 & T \end{bmatrix} \tag{A.4}$$

and Equations 2.21 and 2.22 yield

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{A.5}$$

$$\Gamma = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \tag{A.6}$$

The preceeding calculations are done simply to allow verification of the PHIDEL subroutine in Appendix C. This subroutine converts an **A** and **B** continuous system to a Φ and Γ discrete system. For instance, assume that the system is sampled at f_s = 100 Hertz. This means that T = 0.01. Equations A.5 and A.6 become

$$\Phi = \begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix} \tag{A.7}$$

$$\Gamma = \begin{bmatrix} .00005 \\ .01 \end{bmatrix} \tag{A.8}$$

for the discrete time representation of the second order integrator.

Before proceeding with the OPTCON program, the user is urged to verify that the system is controllable and observable.

b. **Controllability and Reachability**

According to Astrom, a system is controllable [Ref. 5: p. 104] only if "it is possible to find a control sequence such that *the origin* can be reached from any initial state in finite time." Thus, controllability is a necessary condition for the regulator problem. A similar property called reachability is required for the tracking problem. A system is reachable [Ref. 5: p. 104] only if "it is possible to find a control sequence such that *an arbitrary state* can be reached from any initial state in finite time."

—For continuous time systems, the properties of controllability and reachability coincide. That is, either a continuous time system is both controllable and reachable or it is both uncontrollable and unreachable. For a discrete time system, however, controllability does not guarantee reachability. Reachability of a discrete time system does guarantee controllability. The reachability of a discrete system is important because the engineer should not spend a lot of time designing a controller that is physically impossible to implement.

A simple test is performed to check the reachability of a discrete system. The $(n \times n\ell)$ reachability matrix, W_r, for an n^{th} order discrete time system with ℓ control inputs is defined as follows :

$$W_r = \begin{bmatrix} \Gamma & \Phi\Gamma & \Phi^2\Gamma & \ldots & \Phi^{(n-1)}\Gamma \end{bmatrix} \tag{A.9}$$

If the reachability matrix is of rank n, then the system is reachable. In the case of the example problem

$$W_r = \begin{bmatrix} T^2/2 & T^2 \\ 0 & T \end{bmatrix} \qquad (A.10)$$

Taking the determinant of W_r and setting the result equal to zero, the necessary condition for reachability is found to be that $T \neq 0$. Since an infinite sampling frequency is impossible to achieve, the system is reachable and it is reasonable to continue with the problem.

c. **Observability**

In order to take full advantage of the optimal gain schedule, **F**(k), it is necessary that *all* of the states be observable. According to VanLandingham [Ref. 4: p. 308], a discrete time system is completely observable if it is possible to determine the initial state, **x**(0), of the system based on knowledge of the control input, **u**(k), and the output, **y**(k), over a finite number of time intervals. The test for observability closely follows the test for reachability. First, define the (mn × n) observability matrix, W_o, as

$$W_o = \begin{bmatrix} C \\ C\Phi \\ C\Phi^2 \\ \vdots \\ C\Phi^{(n-1)} \end{bmatrix} \qquad (A.11)$$

If the rank of W_o is n, then the system is observable. This implies that all of the states of the system are available for state feedback. If the system is not completely observable, then one or more of its states is not measureable. Either the system must operate without the unobserved states in the feedback path, or an observer must be designed to estimate the unobserved states. In the example problem, the observability matrix is

$$W_o = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & T \\ 0 & 1 \end{bmatrix} \qquad (A.12)$$

This observability matrix is of rank 2 and the system is completely observable. Notice that if the output distribution matrix is

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{A.13}$$

so that only x_1 is observed, the observability matrix becomes

$$W_o = \begin{bmatrix} 1 & 0 \\ 1 & T \end{bmatrix} \tag{A.17}$$

which is of rank 2 provided that $T \neq 0$. However, if only x_2 is observed, then

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{A.15}$$

$$W_o = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{A.16}$$

and the system is not observable regardless of the sampling frequency. A state observer is needed to give an estimate of the x_1 state.

d. Solution Using OPTCON

This section is an introductory guide to OPTCON. The second order integrator problem is used to acquaint the user with the commands, features, and limitations of the program. The messages presented in this section are referred to as "screens" and are surrounded by numbered boxes. Neither the boxes nor the numbers by which they are referenced are actual features of the OPTCON program. They are simply used as devices to make the following discussion more understandable. Messages which are generated by OPTCON appear in standard print. Any responses which represent keyboard entry by the user are shown in *italic* print. If the response is to be *y* for "yes" or *n* for "no", then either uppercase or lowercase letters are acceptable. If the response is to be an integer entry, as in the menu selections, the subprogram COMPARE is called to verify that the user has entered a valid integer. If the response is out of range of the acceptable values, or if the response is not an integer, then the program repeats the message until a valid response is entered by the user.

1. Starting the Program

The user enters OPTCON by typing *optcon* on the command line. The following heading appears

Screen 1

```
OPTCON minimizes the following cost function:

J = MIN  ( X'(N) * H * X(N) + Sum( X'(k) * Q * X(k) + U'(k) * R * U(k)))

The output of the program is the feedback gain matrix, F transpose, (F'),
which, when multiplied by the State Vector (X),
yields a scalar control.(U).


The following recursive equations were derived using dynamic programming,
starting at the terminal time (N) and working backwards:

    (1) F'(k)  = -(DEL'*P(k-1)*PHI)/(DEL'*P(k-1)*DEL + R)        F'(0)=0
    (2) PSI(k) = PHI + DEL*F'(k)                                 PSI(0)=0
    (3) P(k)   = PSI'(k)*P(k-1)*PSI(k) + Q + F'(k)*R*F(k)         P(0)=H
```

2. Entering Initial Information

The first entry required is a problem name. This name is used to identify the output file called OPTFILE which contains all matrices, gain trajectories, and time response trajectories for each run that the user requests during the problem session.

Screen 2

```
First enter the problem identification ( NOT to exceed 20 characters ).

        PROBLEM ID.....second order example
```

Next, the user selects the type of printer that is connected to the operating system. If graph hardcopies are not to be requested during the course of the problem session, then the response to this question has no impact on the operation of OPTCON. If graph hardcopies are to be requested, however, then the answer to this question sets a flag that allows data to be properly formatted for the printer that is being used. Unpredictable results are expected if the user attempts to get graph

hardcopies from a printer that is not selected. In this example, the LaserJet printer is to be selected.

Screen 3

```
Select the type of printer that you are using
        ( Answer    1 or 2 )

        1)  EPSON or THINKJET
        2)  LASERJET

    ANSWER.....2
```

Now the user enters the order, n, of the system. The maximum order which OPTCON can accept is eight due to the limitation of 64 kbytes per segment in the IBM AT microcomputer. The practice problem requires that a *2* be entered here.

Screen 4

```
Enter the ORDER of the system (up to 8).   2
```

3. *Entering the Cost Function*

Next, the number of discrete time stages, N, is entered. This number is limited to 1000 due to the maximum dimension size of the arrays in OPTCON. The user should be aware of the relationship :

$$t_f = N\Delta t \tag{A.18}$$

where

t_f = final time of the process

N = number of stages over which the $\sum$ in Equation 2.23 is to be performed.

Δt = sampling interval

In the example, let $\Delta t = 0.01$ seconds and $t_f = 10$ seconds. This requires $N = 1000$.

Screen 5

```
Enter the NUMBER of TIME INTERVALS (N) over which the cost function
is to be minimized.  (MUST NOT exceed 1000)    1000
```

At this point, the weighting elements of the cost function are to be entered. Assuming that the user wants to initially create a *baseline* solution for the problem, a reasonable starting point is to let all diagonal weighting factors assume a value of unity. The routine to enter the cost function begins with Screen 6.

Screen 6

```
Does cost function (J) include the State TRAJECTORY over all stages ?
              ( Answer    1,2,or 3 )

       1)  YES...Set  Q  equal to the IDENTITY Matrix .
       2)  YES...Each diagonal element of Q will be entered separately.
       3)  NO....Set  Q  equal to the   ZERO   Matrix .

       ANSWER.............
```

Selecting option 1 results in the Q matrix being echoed in Screen 7. The program then advances directly to Screen 11.

Screen 7

```
The states are weighted equally for the TRAJECTORY over all stages.

     The Q Matrix

   1.0000      .0000
    .0000     1.0000
```

Selecting option 2 in response to Screen 6 allows the user to enter values for the diagonal elements of the Q matrix. All off-diagonal elements are automatically set equal to zero. For the sample problem, assume that the user wants $q_{11} = 2.4$ and $q_{22} = 5$. After entering the value for q_{11}, the user is prompted to enter the value for q_{22}. Screen 8 results.

Screen 8

```
Enter the elements of the Q matrix.

(State weighting matrix for TRAJECTORY over all stages)

    Q(1,1) = ?  2.4
    Q(2,2) = ?  5
```

After the user enters all diagonal elements, the matrix is echoed in Screen 9. OPTCON then advances to Screen 11.

Screen 9

```
     The Q Matrix

   2.4000       .0000
    .0000      5.0000
```

Selecting option 3 in response to Screen 6 sets all elements of the Q matrix equal to zero. Screens 10 and 11 follow.

Screen 10

```
The state TRAJECTORY is not included in your cost function.

     The Q Matrix

    .0000       .0000
    .0000       .0000
```

Screen 11 involves a loop which allows the user to change any or all of the elements in the matrix that is currently being processed. This loop is subsequently referred to in this discussion as "the modify routine."

Screen 11

```
Do you want to change any element of the matrix?

  1)  YES...a SINGLE element.
  2)  YES...the ENTIRE Matrix.
  3)  NO

  ANSWER.....
```

Option 1 produces Screen 12 which allows the user to change a single element by identifying the row and column of the element to be changed. The row and column entries *must be* integers *separated by a comma.* Assume that the user wants to change q_{22} so that it equals 3.

Screen 12

```
Which element of the Matrix do you want to Change ?
If I is the ROW and J is the COLUMN,....enter   I,J   2,2
```

The user is then prompted to enter the new value for the element that is to be changed. Screen 13 applies.

Screen 13

```
Q(2,2) =  ? 3
```

If the user entered this situation directly from Screen 7 then the result is Screen 14.

Screen 14

```
     The Q Matrix

   1.0000       .0000
    .0000      3.0000

Any other changes?  (Answer  y or n)
```

If the answer to Screen 14 is *y*, then OPTCON returns to Screen 12 and allows changes to be made to individual elements of the matrix. Once the user is satisfied that that the Q matrix is correct, a *n* is entered in response to Screen 14. At this point, OPTCON is ready to accept information relating to the H matrix. Screen 15 is next.

Screen 15

```
Does cost function (J) include TERMINAL States ?
              ( Answer  1,2,or 3 )

      1)  YES...Set  H  equal to the IDENTITY Matrix .
      2)  YES...Each diagonal element of H will be entered separately.
      3)  NO....Set  H  equal to the   ZERO   Matrix .

     ANSWER.............
```

The program operation at this point is identical to the operation illustrated in Screens 6 through 14. The only difference now is that all of the matrix information applies to the H matrix. Assume that the user has set $h_{11} = 5$ and $h_{22} = 15$. Screen 16 results.

Screen 16

```
     The H Matrix

   5.0000       .0000
    .0000     15.0000

Any other changes?  (Answer  y or n)
```

Since this is the desired H matrix, a *n* is entered and the program advances to the section in which the user is asked to enter the value for R. Assume the desired value is to be 15.7. Screens 17 and 18 result.

Screen 17

```
Enter the value of the scalar R
(Control input weighting factor)

R = ?   15.7
```

Screen 18

```
The scaler R =  15.7
Any changes to R ?  (Answer  y or n)
```

The program echoes the value entered and asks if there are any changes. If there are changes to be made, a *y* response returns the user to Screen 17. A *n* response in Screen 18 indicates that the cost function, J, is now defined completely and Block 1 of Figure A.1 is complete. The program advances to Block 2 of Figure A.1

The user must now indicate if the problem to be solved is in the continuous time domain or in the discrete time domain. Screen 19 applies.

Screen 19

```
If you want to read in A and B matrices for a CONTINUOUS TIME system
 ..................................................Enter 0

If you want to enter PHI and DEL matrices for a DISCRETE TIME system,
 ..................................................Enter 1

ANSWER.....0
```

The sample problem is of the continuous type and a *0* is the appropriate response to Screen 19. Screen 20 follows.

Screen 20

```
You will enter the A and B matrices.
 ........Is this correct ?
```

If a *y* response is entered in Screen 20, then the program advances to Screen 22. Otherwise, the message in Screen 21 appears.

Screen 21

```
You will enter the PHI and DEL matrices.
 ..........Is this correct ?
```

The program toggles between Screens 20 and 21 until the user enters *y* to one of these two options. Assuming that the continuous system is selected, the next section of the program allows the user to enter the **A** and **B** system matrices and the sampling interval, T.

4. Entering the Continuous Time System Parameters

The elements of the **A** matrix are sequentially entered as shown in Screen 22.

Screen 22

```
Enter the elements of the plant matrix--A.

A(1,1) = ?  0
A(1,2) = ?  1
A(2,1) = ?  0
A(2,2) = ?  0
```

Screen 23 echoes the **A** matrix and affords the user an opportunity to make any changes. The modify routine is entered unless the user responds to Screen 23 with a *3*. In the case shown, all entries are correct and a *3* is appropriate.

Screen 23

```
The A Matrix (Plant Matrix)

    .0000     1.0000
    .0000      .0000

Do you want to change any element of the matrix?
     1)  YES...a SINGLE element.
     2)  YES...the ENTIRE Matrix.
     3)  NO

ANSWER.............3
```

The elements of the **B** matrix are sequentially entered as shown in Screen 24.

Screen 24

```
Enter the elements of the distribution matrix--B.

B(1,1) = ? 0
B(1,1) = ? 1
```

Screen 25 echoes the **B** matrix and once again allows the user to enter the modify routine if necessary.

Screen 25

```
The B Matrix (Distribution Matrix)

      .0000
     1.0000

Do you want to change any element of the matrix?
     1)  YES...a SINGLE element.
     2)  YES...the ENTIRE Matrix.
     3)  NO

ANSWER.............3
```

Since no changes are needed, the program now prompts the user to enter the sampling interval, T. The correct answer for the sample problem is entered in Screen 26.

Screen 26

```
Enter the SAMPLE INTERVAL.....DT = ? 0.01
```

As usual, the response is echoed and the user is allowed to make changes until the correct value is entered. Screen 27 applies.

Screen 27

```
The SAMPLE INTERVAL DT =   .0100
Any changes to the SAMPLE INTERVAL ?  (Answer  y or n)  n
```

5. The Optimal Feedback Gains Calculated

The program now has all of the information necessary to calculate the optimal gain schedule. The first step that OPTCON must perform is to convert the **A** and **B** matrices to the corresponding Φ and Γ matrices. The subroutine PHIDEL in Appendix B performs this conversion. The resulting Φ and Γ matrices are not displayed on the monitor. These two matrices are, however, recorded in the OPTFILE listing for the user's convenience. If a discrete time system is initially selected in Screens 19 and 20, then the PHIDEL subroutine is bypassed. In either case, the gain schedule is now calculated using Equations 2.28, 2.29, and 2.30. This completes Block 3 of Figure A.1. As block 4 of Figure A.1 is entered, the user may choose to view the gain schedule in tabular form. Screen 28 applies.

Screen 28

```
Do you want to see the gain schedule table on the screen ?
(Answer   y or n)   y
```

Since the user wishes to view the gain schedule table on the monitor, a *y* is entered in Screen 28. The user should remember two points before choosing to list the gain schedule on the screen :

1. The gain schedule is automatically entered into OPTFILE regardless of the user's response in Screen 28. If the user wants to record the exact values of the

gains, this output file may be examined later using the BROWSE, COPY, EDIT, PRINT, or TYPE commands in DOS.

2. A total of N lines of data are listed on the monitor when tabular output is requested in Screen 28. If N is on the order of several hundred or more, the design procedure is likely to lose momentum due to the lengthy delay involved in sending such a large array to the monitor.

In order to illustrate the form of the data generated, Screen 29 lists a portion of the gain schedule table. Only the first ten time intervals are listed here for brevity. The actual sample problem lists a table with 1000 rows.

Screen 29

```
   NEG      REAL
   TIME     TIME
   STEP     INDEX     F'(1)     F'(2)

*************************************************************************

     1       1000       .0000    -.0100
     2        999      -.0002    -.0200
     3        998      -.0004    -.0300
     4        997      -.0008    -.0400
     5        996      -.0012    -.0500
     6        995      -.0018    -.0600
     7        994      -.0024    -.0700
     8        993      -.0032    -.0800
     9        992      -.0040    -.0900
    10        991      -.0050    -.1000
```

Block 4 of Figure A.1 is now complete and Block 5 is initiated. The next option available to the user is to have OPTCON generate graphs of the optimal gain trajectories. If graphs are not desired, the user may answer *n* in response to Screen 30 and the program advances to Screen 32. If plots of the gain trajectories are desired, then a *y* response is required in Screen 30.

Screen 30

```
Do you want to see the gains plotted ?

(Answer  y or n)  y
```

At this point, the program calls subroutine GRAPH. An internal flag is set so that the gain trajectory plots are sent to the monitor. A separate plot is generated for each gain trajectory. Thus, for an n^{th} order system, there are n separate gain plots produced. As each graph is generated on the screen, a pause is inserted so that the user may conveniently examine each one. Striking any key removes the current graph from the monitor and Screen 31 follows.

Screen 31

```
Do you want a hardcopy of this plot ? ( Answer  y or n )  y
```

If a *n* is entered in Screen 31, then the program begins to generate the next gain trajectory plot for monitor output. By answering *y* in Screen 31, the user will automatically be provided with a hardcopy of the gain trajectory. A single hardcopy graph requires approximately 120 seconds on the Epson printer and approximately 90 seconds on the Laserjet printer. Because of the superior quality of the graphs available from the later, all graphs contained in this thesis are generated on the Laserjet printer. As soon as the hardcopy is complete, OPTCON begins to generate the next gain trajectory plot for monitor output. It is important to note that the gain trajectories are plotted against the *real* time discrete index, k. This means that the first gains *calculated* are those on the *rightmost* edge of the plot while the first gains *implemented* are those on the *leftmost* edge of the plot. Thus, the term "steady state" as it applies to optimal feedback gains refers to the zero-slope property of the *left* side of the gain trajectory plot. The two gain trajectory plots for the example problem are shown in Figures A.3a and A.3b. When all n gain plots have been displayed on the monitor and/or have been printed as hardcopies, the program continues with Screen 32.

Screen 32

```
Do you want to change the NUMBER OF STAGES ?
(Answer  y or n)  n
```

If the user is not satisfied with the initial choice of N, then a new value may be entered at this time by answering *y* in Screen 32. OPTCON presents Screen 5 for this purpose and subsequently returns to the sequence beginning with Screen 28.

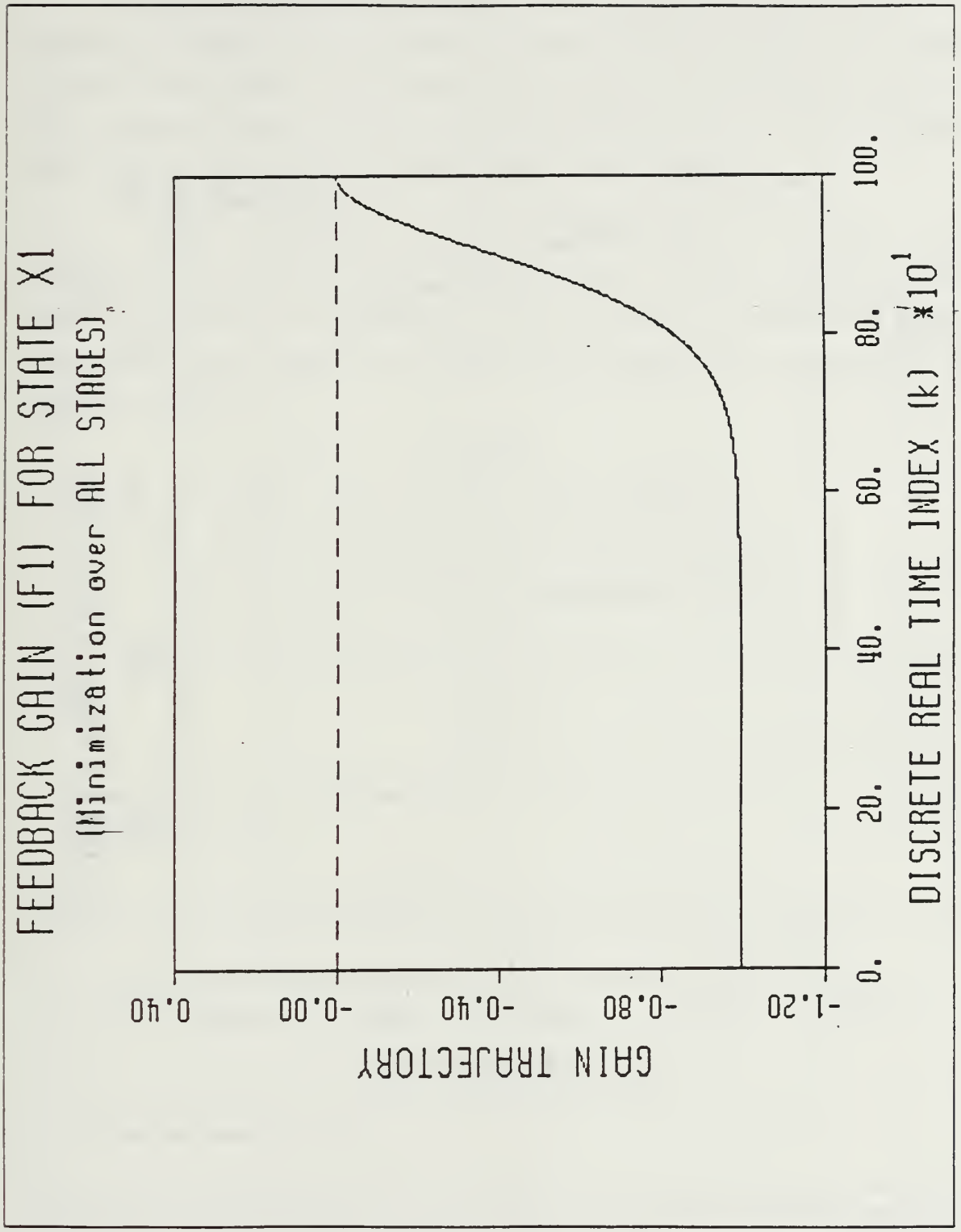

Figure A.3a Gain Trajectory for State x_1

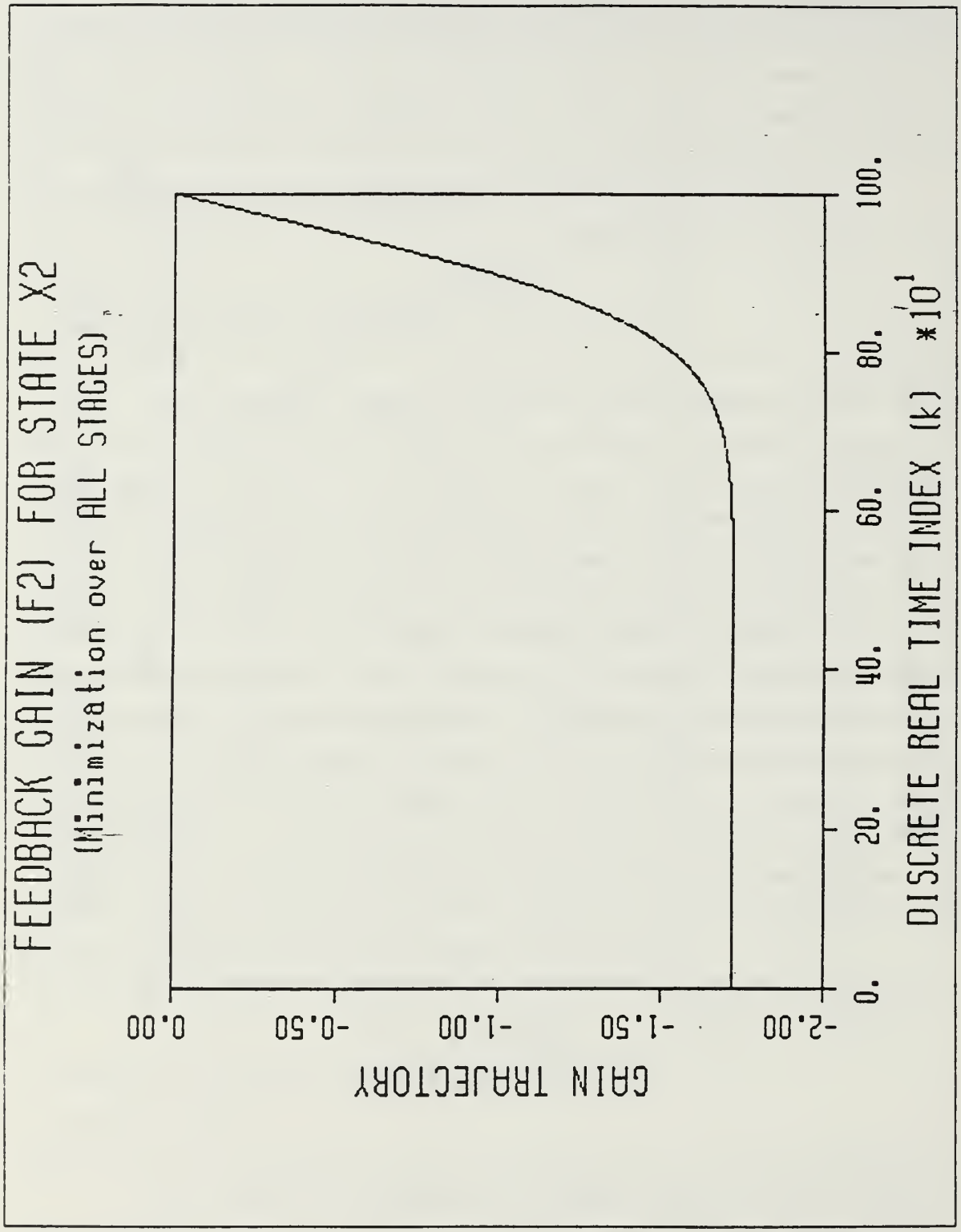

Figure A.3b Gain Trajectory for State x_2

The most likely reason for the user to take advantage of the option in Screen 32 is that the gain trajectories do not reach steady state values in the allotted number of time intervals. By increasing N, the user may be able to force the gains to reach steady state. Since the gain trajectories in Figures A.3a and A.3b demonstrate steady state properties, there is no need to change N in Screen 32.

6. The Time Response

Block 6 in Figure A.1 involves calculation of the time response of the system based on the optimal gains computed in Block 3. The first option available to the user in this section is the phase plane graph of x_1 verses x_2. Screens 33 through 36 represent the program sequence that results when a phase plane is requested with the following constraints :

t_f	=	10 seconds
$x_1(0)$ = 0	=	Initial condition on state x_1
$x_2(0)$ = 0	=	Initial condition on state x_2
r(1) = 1	=	Step forcing function on state x_1
r(2) = 0	=	Ramp forcing function on state x_2

Screen 33

```
Do you want to see a PHASE PLANE of X1 .vs. X2 ?
(Answer  y or n)  y
```

Screen 34

```
For how many seconds ?  10
```

Screen 35

```
Enter the elements of the INITIAL STATE vector  - Xk(0)

X1(0) = ?  0
X2(0) = ?  0
```

Screen 36

```
Enter the elements of the COMMAND INPUT vector-R.

R(1) = ?  1
R(2) = ?  0
```

The next option available is to select the method by which the optimal gains are to be implemented. Two choices are available.

Screen 37

```
Select a gain schedule....( Answer  1 or 2 )

        1)  Use STEADY STATE gains over all steps .
        2)  Use DYNAMIC gains .

        ANSWER.............
```

If the first option is chosen, then the state trajectories are calculated using Equations 2.14 through 2.17 such that the *last* gain matrix calculated, F(N), is substituted into Equation 2.17 during *every* cycle of the iteration process. The user must be aware of this procedure when selecting option 1 in Screen 37 because OPTCON makes no attempt to verify that the gains have indeed reached steady state. If the user selects option 1 when the gains do not exhibit steady state properties, then the solution is *not* optimal. If the gain trajectories do arrive at steady state prior to the N^{th} stage, then selection of option 1 in Screen 37 may be appropriate. The time response obtained in this fashion represents the behavior of the system using a *fixed gain* feedback scheme.

The second option in Screen 37 causes the feedback gains to be dynamically implemented in the *reverse* order that they are calculated. The user is cautioned that such implementation may not yield an acceptable time response. In the example problem, the gains reach steady state after approximately 500 stages. This corresponds to $t_f = 5$ seconds when $\Delta t = 0.01$. Consider the case of a sampled system which has a transient time response longer than 5 seconds. The use of a dynamic gain schedule would be disasterous in this situation. Because the gains progress towards zero as t approaches 5 seconds, the feedback channel is gradually eliminated from the system. The slow system, however, does not have enough time to reach steady state before the feedback gains go to zero. The error signal increases without bound and the system rapidly becomes unstable. Two simple solutions for such a situation are :

1. Increase the number of time intervals, N. This causes the steady state portion of the dynamic gain schedule to become more predominate.
2. Implement steady state gains instead of a dynamic gain schedule.

After a gain schedule is selected in Screen 37, OPTCON begins to compute and save the state trajectories for x_1 and x_2 using Equations 2.14 through 2.17. The message in Screen 38 informs the user that the program is still executing.

Screen 38

```
Calculating Plotting Data
```

After the state trajectories are computed, Screen 39 prompts the user.

Screen 39

```
READY TO DISPLAY DRAWING
Strike any key to continue.
```

The monitor is cleared upon any keystroke and the x_1 verses x_2 phase plane subsequently appears on the screen. The graph remains on the screen until the user strikes any key. The monitor then clears and Screen 40 appears.

Screen 40

```
Do you want a hardcopy of this plot ? ( Answer  y or n )  y
```

If a *n* is entered in Screen 40, then the program advances to Screen 41. Otherwise, the message in Screen 38 reappears. After a short delay, a hardcopy graph of the phase plane is automatically generated on the printer. See Figure A.4. The program then advances to Screen 41.

Screen 41

```
Do you want to see a time response of your system ?
(Answer  y or n)
```

If a *n* is entered in Screen 41, then the first run of OPTCON is complete. The program advances to Screen 44. If a *y* is entered in Screen 41, then the program prompts the user to enter parameters for the time response. Refer to Screens 34 through 37. It is not required that the user enter the same information for the time response that was entered for the phase plane. OPTCON recomputes the time response on *every* run. It is suggested, however, that the user carefully note the parameters that are entered for each run. Initial conditions and command inputs are recorded in the OPTFILE but this information does not appear on the graphs. After the gain schedule is selected in Screen 37, OPTCON begins to compute and save the state trajectories. When the calculations are complete, Screen 42 appears.

Screen 42

```
Do you want to see the time response table on the screen ?
(Answer  y or n)
```

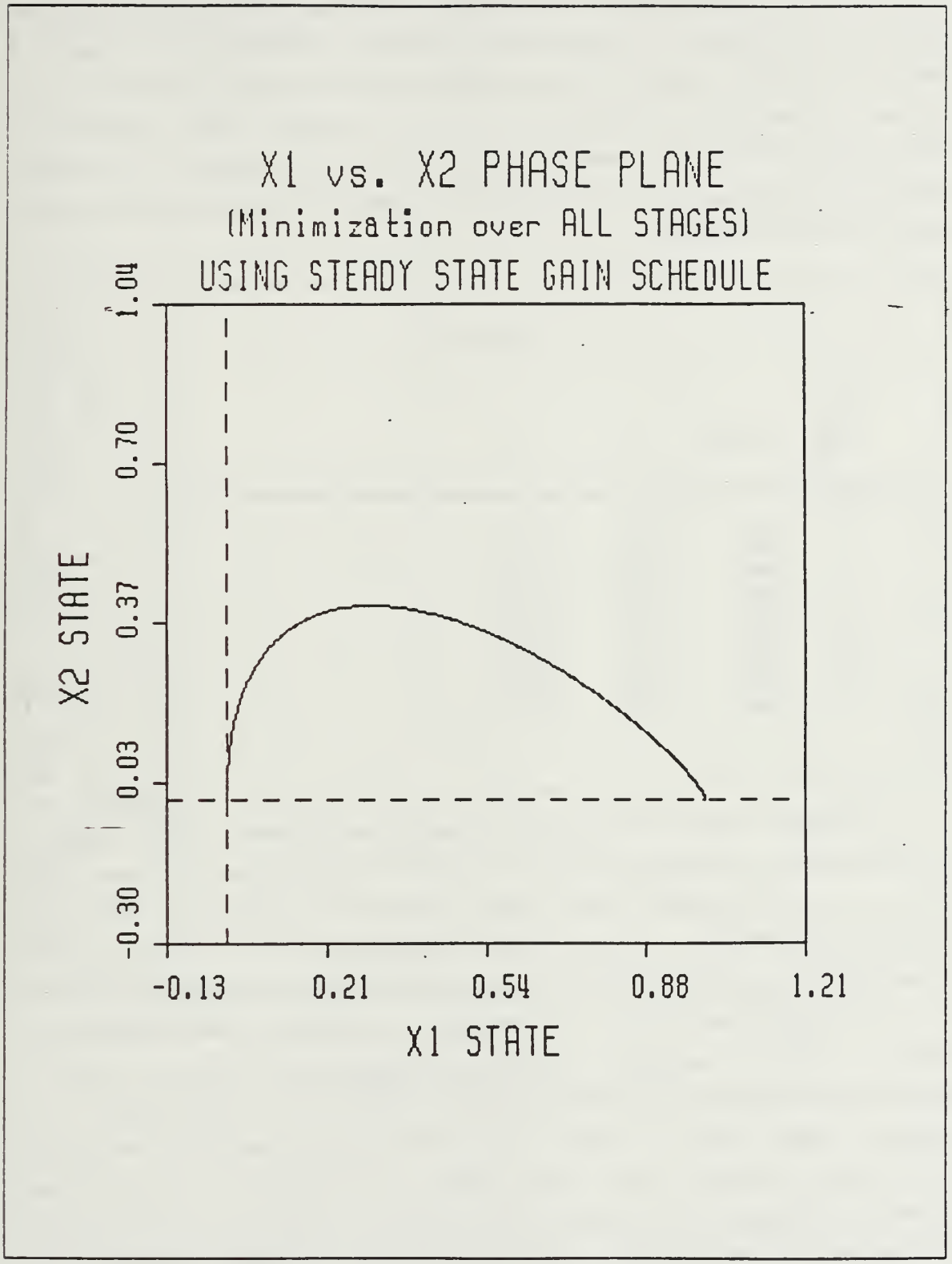

Figure A.4 Phase Plane for Second Order Integrator Example

A *n* response causes the program to begin generating data for the time response plots. The user is cautioned that answering *y* in Screen 42 may result in a lengthy delay as the N rows of data are scrolled onto the monitor. The option to view this data on the monitor exists so that the user may gather exact numerical data without exiting OPTCON to examine the OPTFILE. A short segment of the tabular data appears in Screen 43. In the case of the example problem, this table continues until 1000 rows are displayed.

Screen 43

REAL TIME INDEX	REAL TIME	x(1)	X(2)
1	.0100	.0000	.0000
2	.0200	.0000	.0099
3	.0300	.0002	.0197
4	.0400	.0004	.0292
5	.0500	.0008	.0386
6	.0600	.0012	.0479
7	.0700	.0017	.0570
8	.0800	.0024	.0659
9	.0900	.0031	.0746
10	.1000	.0038	.0832

When the last row of the state trajectory table appears on the monitor, or if tabular output is not selected in Screen 42, then OPTCON begins to generate data for the state trajectory plots. Each state is plotted verses real time. During the calculations, the messages in Screens 38 and 39 prompt the user. State x_1 is plotted first and the n^{th} state is plotted last. The user may examine each individual graph on the monitor. By striking any key, the user clears the graph from the screen and the message in Screen 40 reappears. If the user does not desire a hardcopy, then a *n* response allows the program to process data for the next time response graph. If a hardcopy is desired, then a *y* is entered in Screen 40 and the procedure follows exactly as before. See typical time response plots in Figures A.5a and A.5b. After all n states are plotted, the program completes Block 8 in Figure A.1. The first run of OPTCON is now complete and the user must answer *y* in Screen 44 in order to remain in the program. If a *n* is entered in Screen 44, then execution terminates and the user is immediately returned to the DOS environment.

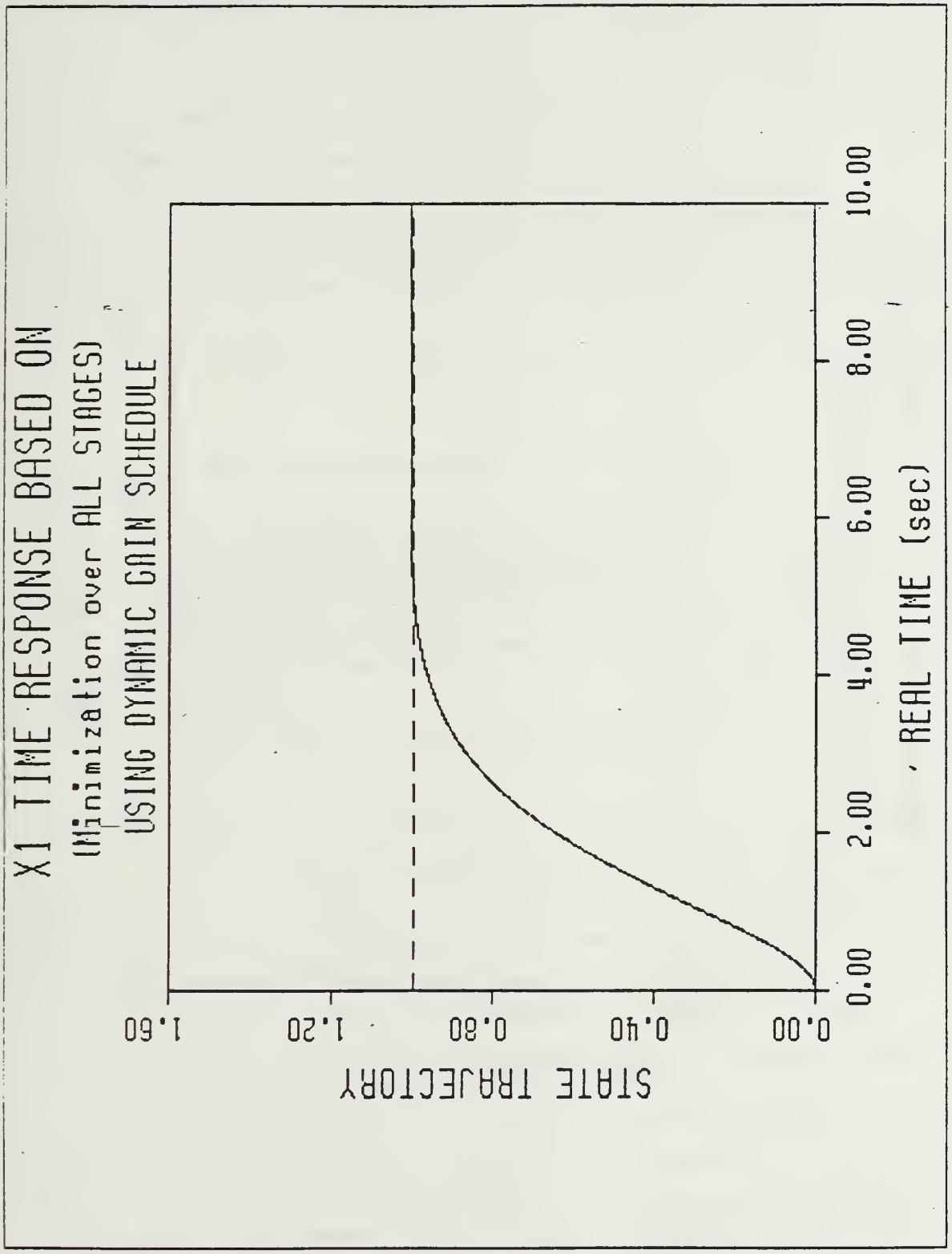

Figure A.5a State x_1 Time Response for Second Order Integrator

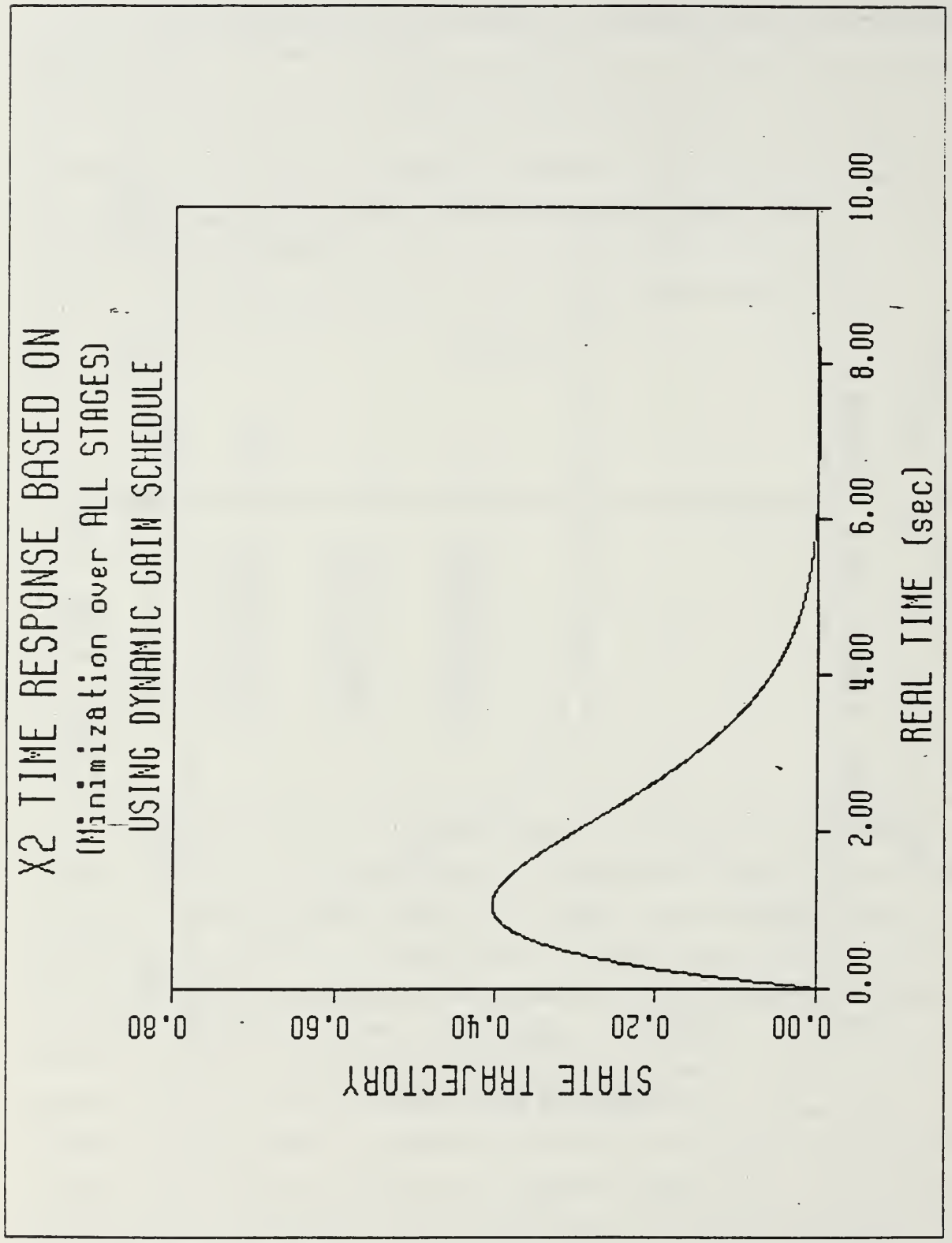

Figure A.5b State x_2 Time Response for Second Order Integrator

Screen 44

```
This concludes the optimal control program (OPTCON).

Do you want to run the program again?  (Answer   y or n)   y
```

Assuming that the user desires to remain in OPTCON, a *y* is entered in Screen 44. The next section of the program is referred to as "the main menu" and is demonstrated in Screen 45.

Screen 45

```
SELECT ONE OF THE FOLLOWING OPTIONS:


 1) Change the NUMBER of STAGES.....................N
 2) Change the TERMINAL state weighting matrix.....H
 3) Change the TRAJECTORY state weighting matrix...Q
 4) Change the CONTROL weighting factor............R
 5) Change the present A and B matrices
 6) Change the SAMPLE INTERVAL.....................DT
 7) Change the present PHI and DEL matrices
 8) Input an entirely NEW SYSTEM
 9) NO CHANGES...RUN
10) EXIT the program

SELECTION...( MUST be a number between 1 and 10 )......
```

It is not necessary to describe in detail the operation of the main menu. The user should enter the integer value that applies to the particular modification to be made. If one of the first seven options is selected, the program responds as follows :

1. Echo the current value(s) of the parameter(s) to be modified.
2. Allow the user to keep or modify the selected parameter(s).
3. Return to the main menu for further modification, continued execution, or termination of the program.

If option 8 in the main menu is selected, then OPTCON returns to Screen 4 and allows the user to enter new information for all parameters. In this case, no previous values are remembered by the program and execution proceeds just as if this is the first run. The OPTFILE, however, retains all information from any previous runs.

If option 9 is selected in the main menu, then the current values for all system and cost function parameters are written into OPTFILE to signal the start of a new run. Program execution begins with the gain calculation sequence represented by Block 3 in Figure A.1. Screen 28 applies. The user may rapidly skip through the intermediate steps of the program by answering *n* to several consecutive questions. For instance, suppose that the user changes a single parameter by selecting one of the first seven options in the main menu. In order to determine the effect of the changed parameter on the time response of the system, the following sequence of messages and responses is used.

Screen 46

```
SELECT ONE OF THE FOLLOWING OPTIONS:

 1) Change the NUMBER of STAGES.....................N
 2) Change the TERMINAL state weighting matrix.....H
 3) Change the TRAJECTORY state weighting matrix...Q
 4) Change the CONTROL weighting factor............R
 5) Change the present A and B matrices
 6) Change the SAMPLE INTERVAL.....................DT
 7) Change the present PHI and DEL matrices
 8) Input an entirely NEW SYSTEM
 9) NO CHANGES...RUN
10) EXIT the program

  SELECTION...( MUST be a number between 1 and 10 ).....9

Do you want to see the gain schedule table on the screen ?

(Answer   y or n)  n

Do you want to see the gains plotted ?

(Answer   y or n)  n

Do you want to see a PHASE PLANE of X1 .vs. X2 ?

(Answer   y or n)  n

Do you want to see a time response of your system ?

(Answer   y or n)  y
```

At this point, the user may examine the system time response and evaluate the impact of the newly modified parameter.

By selecting option 10 in the main menu, the user is allowed to exit the program. The message in Screen 44 reappears as a safety mechanism to prevent inadvertant ejection from the program. If *y* is entered in Screen 44, then the main menu reappears and program execution continues. Otherwise, the program terminates and control is returned to DOS.

7. OPTCON Summary

The OPTCON program is designed specifically so that the user can easily modify problem parameters and rapidly obtain information about the effects of those changes. Tabular and graphical information is available both on the monitor and in hardcopy form. In an effort to make the program user-friendly, four techniques are employed :

1. Menu driven options prevail.
2. User input is screened for valid format.
3. User inputs are echoed on the monitor.
4. All data is written into an external file for later examination.

The OPTCON program is quite useful as an interactive design tool for optimal control systems. Extensive use is made of its assets during the design of an optimal controller for the AROD in chapter three.

APPENDIX B
OPTCON MAIN PROGRAM LISTING

The following code is written in MICROSOFT Fortran and is intended to be used on an IBM compatible system. This is the main program for OPTCON and must be linked with the subroutines found in Appendices C and D. In addition, the Fortran, Math, and PLOT88 libraries must be linked during the creation of an executable file.

```
$NOfloatcalls
$NOdebug
C
C
C LL63.FOR         LAST MOD 12JULY87        OK   SDL
C                                           NEW selective state plotting
C                                           NEW state table formatting
C
      COMMON /BLK1/ A,B,PHI,DEL
      COMMON /BLK2/ BEGTIM,FINTIM,NPTS,
     +              XNAML,YNAML,PNAM1L,PNAM2L,PNAM3L
      COMMON /BLK3/ VTIME,VTIMSS,VY,VYSS,VXXSS,VXYSS
      COMMON /BLK4/ KFINAL,NSTAGE,NSTP1,ORDERN,GNSKED,USERGN,FNEG,
     +              INPUT,DT,AVG,AVG2,MAXVAL,NINPTS
      COMMON /BLK5/ XNAME,YNAME,PNAME1,PNAME2,PNAME3
      INTEGER*2 OPTION,ORDERN,IGOOD,CODE,NSTAGE,NSTP1,KFINAL,KPRIME,
     +          GNSKED,NPTS,IOPORT,MODEL,XNAML,YNAML,NCHAR1,NCHAR2,
     +          NCHAR3,STVAR,I,J,SKIP,OK,SYSTEM,GAIN,DTFLAG,PLTYPE,
     +          CHNGN,SCREEN,NINPTS,NINPP1,ORDNP1,GAINCH,GNSKD3,
     +          STPLOT,PLOTCH
      REAL*4 PSI(8,8),P(8,8),FTRAN(8),GM(8,8),
     +       FM(8),EM(8),HM(8,8),DEL(8,2),
     +       PHI(8,8),A(8,8),B(8,2),Q(8,8),
     +       H(8,8),XK0(8,1),XK(8,1),XKP1(8,1),DELINP(8,1),INPUT(8,1),
     +       PHIEQX(8,1),PHIEQ(8,8),DELROW(8,8),ROWF(8,8),FNEG(1000,8),
     +       VY(1002),VTIME(1002),TFTEMP,TFINAL,DT,TIME,PNAM1L,PNAM2L,
     +       PNAM3L,VYSS(9),VTIMSS(9),VXXSS(9),VXYSS(9),AVG(8),AVG2(8),
     +       MAXTIM,MAXVAL(8),USERGN(8,8)
      CHARACTER*2  TEMP
      CHARACTER*3  ANS
      CHARACTER*20 NAME
      CHARACTER*30 XNAME,YNAME
      CHARACTER*51 PNAME1,PNAME2,PNAME3
      CHARACTER*5  HDG(8)
      CHARACTER*4  HDG2(8)
      HDG(1)  = 'F''(1)'
      HDG(2)  = 'F''(2)'
      HDG(3)  = 'F''(3)'
      HDG(4)  = 'F''(4)'
      HDG(5)  = 'F''(5)'
      HDG(6)  = 'F''(6)'
      HDG(7)  = 'F''(7)'
      HDG(8)  = 'F''(8)'
      HDG2(1) = 'X(1)'
      HDG2(2) = 'X(2)'
      HDG2(3) = 'X(3)'
      HDG2(4) = 'X(4)'
      HDG2(5) = 'X(5)'
      HDG2(6) = 'X(6)'
      HDG2(7) = 'X(7)'
```

```
      HDG2(8) = 'X(8)'
C
      OPEN(9,FILE='OPTFILE',STATUS='NEW')
C
C
C*************************************************************************
C**************** PRINT OPTCON HEADING and INPUT PROBLEM ID ************
C*************************************************************************
C
      WRITE(*,2000)
      WRITE(*,2010)
      PAUSE
      WRITE(*,2015)
      READ (*,2020,END=1530)NAME
C
C*************************************************************************
C*****************    HEADING INFO FOR THE OUTPUT FILE       *************
C*************************************************************************
C
      WRITE(9,2030)
      WRITE(9,2040)
      WRITE(9,2050)NAME
      WRITE(9,2030)
C
C*************************************************************************
C****************      ENTER PLOTTER/PRINTER MODEL TYPE     *************
C*************************************************************************
C
    5 WRITE(*,2055)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,2,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 5
      IF(IGOOD .EQ. 1)THEN
          IOPORT = 0
          MODEL  = 1
      ELSE
          IOPORT = 0
          MODEL  = 60
      ENDIF
C
C*************************************************************************
C*************      INITIALIZE B ,DEL,USERGN MATRICES        *************
C*************************************************************************
C
      DO 6 I = 1,8
          DO 6 J = 1,8
               B(I,J)      = 0.0
               DEL(I,J)    = 0.0
               USERGN(I,J) = 0.0
  6   CONTINUE
C
C*************************************************************************
C****************      ENTER THE ORDER OF THE SYSTEM        *************
C*************************************************************************
C
C****************      RESET FINAL ,GNSKED,GAINCH,GNSKD3    *************
   10 FINAL = 0
      GNSKED = 1
      GAINCH = 0
      GNSKD3 = 0
      WRITE(*,2060)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,8,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 10
      ORDERN = IGOOD
      ORDNP1 = IGOOD + 1
C
C*************************************************************************
C****************      ENTER THE NUMBER OF CONTROL INPUTS   *************
C*************************************************************************
```

```
C
   15 WRITE(*,2075)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,8,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 15
      NINPTS = IGOOD
      NINPP1 = IGOOD + 1
C
C****************                    ECHO     NINPTS            ************
C
      WRITE(*,2076)NINPTS
C
C****************      MODIFY NINPTS  IF NEEDED            ************
C
   16 WRITE(*,2077)
      READ (*,2190)ANSWER
         IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 17
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 15
         GOTO 16
   17 CONTINUE
C
C******  SKIP COST FUNCTION ENTRY IF NUMBER OF CONTROLS .GT. 1   *******
C
      IF(NINPTS .GT. 1) THEN
          GNSKED = 3
          GOTO 340
      ENDIF
C*************************************************************************
C****************     ENTER THE NUMBER OF TIME INTERVALS   ************
C*************************************************************************
C
   20 WRITE(*,2080)
      READ (*,*)NSTAGE
      IF(NSTAGE .GT. 1000)GOTO 20
      NSTP1 = NSTAGE + 1
      IF(CHNGN .EQ. 1)GOTO 780
      IF(FINAL .EQ. 1)GOTO 1520
C
C*************************************************************************
C****************           INPUT THE   Q   MATRIX          ************
C*************************************************************************
C
   30 LOOP = 0
      WRITE(*,2090)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 30
      OPTION = IGOOD
      GOTO(40,50,60) OPTION
      GOTO 30
   40 WRITE(*,2100)
      GOTO 80
   50 WRITE(*,2110)
      GOTO 80
   60 WRITE(*,2120)
   80 DO 90 I = 1,ORDERN
         DO 90 J = 1,ORDERN
            IF(I .EQ. J) THEN
               IF(OPTION .EQ. 1) Q(I,J) = 1.0
               IF(OPTION .EQ. 3) Q(I,J) = 0.0
               IF(OPTION .EQ. 2) THEN
                  WRITE(*,2130)I,J
                  READ (*,*)Q(I,J)
               ENDIF
            ELSE
               Q(I,J) = 0.0
            ENDIF
   90 CONTINUE
C
C****************           ECHO THE   Q   MATRIX          ************
```

```
C
  100 CONTINUE
      WRITE(*,2140)
      DO 110 I=1,ORDERN
  110 WRITE(*,2150)(Q(I,J),J=1,ORDERN)
      IF(LOOP .EQ. 1) GOTO 30
C
C****************      MODIFY THE   Q   MATRIX IF NEEDED    ************
C
  120 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 120
      OPTION = IGOOD
      GOTO(130,50,160)OPTION
       GOTO 120
C
C****************  CHANGE ONE ELEMENT OF THE   Q   MATRIX   ************
C
  130 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.ORDERN)GOTO 130
      WRITE(*,2130)I,J
      READ (*,*)Q(I,J)
      WRITE(*,2140)
      DO 140 I=1,ORDERN
  140 WRITE(*,2150)(Q(I,J),J=1,ORDERN)
  150 WRITE(*,2180)
      READ (*,2190)ANSWER
         IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 160
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 130
         GOTO 150
  160 IF(FINAL.EQ.1)GOTO 1520
C
C*************************************************************************
C****************           INPUT THE   H   MATRIX          ************
C*************************************************************************
C
  170 LOOP = 0
      WRITE(*,2200)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 170
      OPTION = IGOOD
      GOTO(180,190,200) OPTION
      GOTO 170
  180 WRITE(*,2210)
      GOTO 210
  190 WRITE(*,2220)
      GOTO 210
  200 WRITE(*,2230)
  210 DO 220 I = 1,ORDERN
          DO 220 J = 1,ORDERN
              IF(I .EQ. J) THEN
                  IF(OPTION .EQ. 1) H(I,J) = 1.0
                  IF(OPTION .EQ. 3) H(I,J) = 0.0
                  IF(OPTION .EQ. 2) THEN
                      WRITE(*,2240)I,J
                      READ (*,*)H(I,J)
                  ENDIF
              ELSE
                  Q(I,J) = 0.0
              ENDIF
  220 CONTINUE
C
C****************            ECHO THE   H   MATRIX          ************
C
  230 CONTINUE
      WRITE(*,2250)
      DO 240 I=1,ORDERN
```

```
  240 WRITE(*,2150)(H(I,J),J=1,ORDERN)
      IF(LOOP .EQ. 1) GOTO 170
C
C****************      MODIFY THE   H   MATRIX IF NEEDED    ************
C
  250 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 250
      OPTION = IGOOD
      GOTO(260,190,290)OPTION
        GOTO 250
C
C****************  CHANGE ONE ELEMENT OF THE   H   MATRIX   ************
C
  260 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.ORDERN)GOTO 260
      WRITE(*,2240)I,J
      READ (*,*)H(I,J)
      WRITE(*,2250)
      DO 270 I=1,ORDERN
  270 WRITE(*,2150)(H(I,J),J=1,ORDERN)
  280 WRITE(*,2180)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 290
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 260
        GOTO 280
  290 IF(FINAL.EQ.1)GOTO 1520
C
C*******************************************************************
C****************                INPUT    R                ************
C*******************************************************************
C
  300 WRITE(*,2260)
      READ (*,*)R
C
C****************                ECHO     R                ************
C
  310 WRITE(*,2270)R
C
C****************         MODIFY    R     IF NEEDED         ************
C
  320 WRITE(*,2280)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 330
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 300
        GOTO 320
  330 IF(FINAL.EQ.1)GOTO 1520
C
C*******************************************************************
C****************      CHOOSE TO ENTER EITHER A             ************
C****************    CONTINUOUS TIME SYSTEM OR A            ************
C****************        DISCRETE TIME SYSTEM               ************
C*******************************************************************
C
  340 WRITE(*,2290)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,0,1,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 340
      SYSTEM = IGOOD
C
      IF(SYSTEM)350,350,360
  350   WRITE(*,2300)
        READ (*,2190)ANSWER
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 370
        SYSTEM = 1
  360   WRITE(*,2310)
        READ (*,2190)ANSWER
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 590
```

```
          SYSTEM = 0
          GOTO 350
C
C*************************************************************************
C*****************          INPUT THE   A   MATRIX          ************
C*************************************************************************
C
  370 WRITE(*,2320)
      DO 380 I=1,ORDERN
         DO 380 J=1,ORDERN
            WRITE(*,2330)I,J
            READ (*,*)A(I,J)
  380 CONTINUE
C
C*****************   DO NOT ALLOW CHANGES TO    A  and  B    ************
C*****************   IF A DISCRETE TIME SYSTEM WAS ENTERED   ************
C
  390 CONTINUE
      IF(SYSTEM .EQ. 1)THEN
          WRITE(*,2335)
          READ (*,2070)TEMP
          GOTO 1520
      ENDIF
C
C*****************            ECHO THE   A   MATRIX          ************
C
      WRITE(*,2340)
      DO 400 I=1,ORDERN
  400 WRITE(*,2150)(A(I,J),J=1,ORDERN)
C
C*****************      MODIFY THE   A   MATRIX IF NEEDED    ************
C
  410 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 410
      OPTION = IGOOD
      GOTO(420,370,450)OPTION
         GOTO 410
C
C*****************  CHANGE ONE ELEMENT OF THE   A   MATRIX   ************
C
  420 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.ORDERN)GOTO 420
      WRITE(*,2330)I,J
      READ (*,*)A(I,J)
      WRITE(*,2340)
      DO 430 I=1,ORDERN
  430 WRITE(*,2150)(A(I,J),J=1,ORDERN)
  440 WRITE(*,2180)
      READ (*,2190)ANSWER
         IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 450
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 420
         GOTO 440
  450 IF(FINAL.EQ.1)GOTO 480
C
C*************************************************************************
C*****************          INPUT THE   B   MATRIX          ************
C*************************************************************************
C
  460 WRITE(*,2350)
      DO 470 I=1,ORDERN
         DO 470 J = 1,NINPTS
            WRITE(*,2360)I,J
            READ (*,*)B(I,J)
  470 CONTINUE
C
C*****************            ECHO THE   B   MATRIX          ************
C
```

```
  480 CONTINUE
      WRITE(*,2370)
      DO 490 I=1,ORDERN
  490 WRITE(*,2150)(B(I,J),J=1,NINPTS)
C
C****************      MODIFY THE   B   MATRIX IF NEEDED    ************
C
  500 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 500
      OPTION = IGOOD
      GOTO(510,460,540)OPTION
        GOTO 500
C
C****************  CHANGE ONE ELEMENT OF THE   B   MATRIX   ************
C
  510 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.NINPTS)GOTO 510
      WRITE(*,2360)I,J
      READ (*,*)B(I,J)
      WRITE(*,2370)
      DO 520 I=1,ORDERN
  520 WRITE(*,2150)(B(I,J),J=1,NINPTS)
  530 WRITE(*,2180)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 540
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 510
        GOTO 530
  540 IF(FINAL.EQ.1)GOTO 1520
C
C*****************************************************************************
C****************        INPUT THE SAMPLE TIME....DT        ************
C*****************************************************************************
C
  550 WRITE(*,2380)
      READ (*,*)DT
      DTFLAG = 1
C
C****************   IF A DISCRETE TIME SYSTEM WAS ENTERED    ************
C****************   AND NO VALUE FOR DT HAS BEEN ENTERED     ************
C****************         THEN PRINT OUT A MESSAGE           ************
C
  560 IF(DTFLAG .EQ. 0)THEN
          WRITE(*,2385)
          READ (*,2070)TEMP
          GOTO 1520
      ENDIF
C
C****************         ECHO THE SAMPLE TIME....DT         ************
C
      WRITE(*,2390)DT
C
C****************     MODIFY THE SAMPLE TIME IF NEEDED       ************
C
  570 WRITE(*,2400)
      READ (*,2190)ANSWER
          IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 580
          IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 550
          GOTO 570
C
C*****************************************************************************
C****************  CONVERT  A and B   TO   PHI and DEL        ************
C*****************************************************************************
C
  580 IF(SYSTEM .EQ. 0) THEN
          CALL PHIDEL(DT,ORDERN,NINPTS)
      ENDIF
      IF(FINAL.EQ.1)GOTO 1520
```

```
      GOTO 780
C
C*************************************************************************
C*****************          INPUT THE   PHI   MATRIX          ************
C*************************************************************************
C
  590 CONTINUE
      DTFLAG = 0
  600 WRITE(*,2410)
      DO 610 I=1,ORDERN
         DO 610 J=1,ORDERN
            WRITE(*,2420)I,J
            READ (*,*)PHI(I,J)
  610 CONTINUE
C
C*****************   DO NOT ALLOW CHANGES TO   PHI and DEL   *************
C*****************  IF A CONTINUOUS TIME SYSTEM WAS ENTERED  *************
C
  620 CONTINUE
      IF(SYSTEM .EQ. 0)THEN
         WRITE(*,2425)
         READ (*,2070)TEMP
         GOTO 1520
      ENDIF
C
C*****************           ECHO THE   PHI   MATRIX          *************
C
      WRITE(*,2430)
      DO 630 I=1,ORDERN
  630 WRITE(*,2150)(PHI(I,J),J=1,ORDERN)
C
C*****************     MODIFY THE   PHI   MATRIX IF NEEDED   *************
C
  640 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 640
      OPTION = IGOOD
      GOTO(650,600,680)OPTION
         GOTO 640
C
C***************** CHANGE ONE ELEMENT OF THE   PHI   MATRIX  *************
C
  650 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.ORDERN)GOTO 650
      WRITE(*,2420)I,J
      READ (*,*)PHI(I,J)
      WRITE(*,2430)
      DO 660 I=1,ORDERN
  660 WRITE(*,2150)(PHI(I,J),J=1,ORDERN)
  670 WRITE(*,2180)
      READ (*,2190)ANSWER
         IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 680
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 650
         GOTO 670
  680 IF(FINAL.EQ.1)GOTO 710
C
C*************************************************************************
C*****************          INPUT THE   DEL   MATRIX          ************
C*************************************************************************
C
  690 WRITE(*,2440)
      DO 700 I=1,ORDERN
         DO 700 J=1,NINPTS
            WRITE(*,2450)I,J
            READ (*,*)DEL(I,J)
  700 CONTINUE
C
C*****************           ECHO THE   DEL   MATRIX          *************
```

```
C
  710 CONTINUE
      WRITE(*,2460)
      DO 720 I=1,ORDERN
  720 WRITE(*,2150)(DEL(I,J), J=1,NINPTS)
C
C****************     MODIFY THE   DEL   MATRIX IF NEEDED   ************
C
  730 WRITE(*,2160)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,3,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 730
      OPTION = IGOOD
      GOTO(740,690,770)OPTION
        GOTO 730
C
C**************** CHANGE ONE ELEMENT OF THE   DEL   MATRIX  ************
C
  740 WRITE(*,2170)
      READ (*,*)I,J
      IF(I.LT.1 .OR. I.GT.ORDERN .OR. J.LT.1 .OR. J.GT.NINPTS)GOTO 740
      WRITE(*,2450)I,J
      READ (*,*)DEL(I,J)
      WRITE(*,2460)
      DO 750 I=1,ORDERN
  750 WRITE(*,2150)(DEL(I,J),J=1,NINPTS)
  760 WRITE(*,2180)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 770
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 740
        GOTO 760
  770 IF(FINAL.EQ.1)GOTO 1520
C
C*************************************************************************
C****************   WRITE ALL CURRENT INFORMATION TO THE    ************
C****************              OUTPUT FILE                  ************
C*************************************************************************
C
  780 FINAL = 0
      WRITE(9,2030)
      WRITE(9,2470)ORDERN
      WRITE(9,2475)NINPTS
    --IF(GNSKED .EQ. 3) GOTO 805
      WRITE(9,2480)NSTAGE
      WRITE(9,2140)
      TRACEQ = 0.0
      DO 790 I=1,ORDERN
         TRACEQ = TRACEQ + Q(I,I)
  790   WRITE(9,2490)(Q(I,J),J=1,ORDERN)
      WRITE(9,2250)
      DO 800 I=1,ORDERN
  800 WRITE(9,2490)(H(I,J),J=1,ORDERN)
      WRITE(9,2270)R
  805 IF(SYSTEM) 810,810,840
  810 WRITE(9,2340)
        DO 820 I=1,ORDERN
  820   WRITE(9,2490)(A(I,J),J=1,ORDERN)
        WRITE(9,2370)
        DO 830 I=1,ORDERN
  830   WRITE(9,2490)(B(I,J),J=1,NINPTS)
        WRITE(9,2390)DT
  840 WRITE(9,2430)
        DO 850 I=1,ORDERN
  850   WRITE(9,2490)(PHI(I,J),J=1,ORDERN)
        WRITE(9,2460)
        DO 860 I=1,ORDERN
  860   WRITE (9,2490)(DEL(I,J),J=1,NINPTS)
      WRITE(9,2030)
      IF(GNSKED .EQ. 3) THEN
C********      NO OPTIMAL GAINS ARE TO BE CALCULATED      ************
```

```
          GOTO 1010
        ENDIF
        IF(TRACEQ)870,870,880
  870    WRITE(9,2500)
          PNAME2= '(Minimum TERMINAL STATES Control)'
          PNAM2L= 33
          GO TO 890
  880 WRITE(9,2510)
          PNAME2= '(Minimization over ALL STAGES)'
          PNAM2L= 30
C
C*************************************************************************
C****************        INITIALIZE MATRICES PRIOR TO       ************
C****************          CALCULATING OPTIMAL GAINS        ************
C*************************************************************************
C
  890 CONTINUE
        DO 900 I=1,ORDERN
          EM(I) = 0.0
          FM(I) = 0.0
          DO 900 J=1,ORDERN
              GM(I,J) = 0.0
              HM(I,J) = 0.0
  900 P(I,J)=H(I,J)
C
C*********  DO YOU WANT TO SEE THE GAINS TABLE ON THE SCREEN ? *********
C
        WRITE(*,2515)
        READ (*,2190)ANSWER
          IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')SCREEN = 0
          IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')SCREEN = 1
C
C*****************      PRINT HEADING FOR OUTPUT TABLE       ************
C*****************                OPTIMAL GAINS              ************
C
        IF(SCREEN .EQ. 1)THEN
           WRITE(*,2520)(HDG(I),I=1,ORDERN)
           WRITE(*,2030)
        ENDIF
        WRITE(9,2520)(HDG(I),I=1,ORDERN)
        WRITE(9,2030)
C
C*************************************************************************
C*****************     LOOP TO ITERATE THE RICATI EQUATIONS    ***********
C*************************************************************************
C
        DO 1000 KK=1,NSTAGE
          KREAL = NSTP1 - KK
          DEN=0.0
          DO 910 I=1,ORDERN
              DO 910 J=1,ORDERN
  910    EM(I) = EM(I) + DEL(J,1) * P(J,I)
          DO 930 I=1,ORDERN
              DO 920 J=1,ORDERN
  920       FM(I) = FM(I) + EM(J) * PHI(J,I)
  930    DEN = DEN + EM(I) * DEL(I,1)
          DEN = DEN + R
C
C********* ENSURE THAT THE DENOMINATOR DOES NOT GO TO ZERO ***********
C
        IF( DEN .EQ. 0 )THEN
            WRITE(*,2530)KK-1
            WRITE(9,2530)KK-1
            NSTAGE = KK - 1
            GOTO 1007
        ENDIF
C
C*****************     CALCULATE OPTIMAL GAINS FOR THIS STEP  **********
C
          DO 940 I=1,ORDERN
```

```
            FTRAN(I)    = -FM(I)/DEN
            FNEG(KK,I) =  FTRAN(I)
            FM(I)       =  0.0
  940   EM(I) = 0.0
C
C*****************      PRINT OPTIMAL GAINS FOR THIS STEP    ***********
C
        IF(SCREEN .EQ. 1)THEN
            IF(ORDERN .GT. 4) THEN
               WRITE(*,2540)KK,KREAL,(FTRAN(I),I=1,ORDERN)
            ELSE
               WRITE(*,2541)KK,KREAL,(FTRAN(I),I=1,ORDERN)
            ENDIF
        ENDIF
        IF(ORDERN .GT. 4) THEN
            WRITE(9,2540)KK,KREAL,(FTRAN(I),I=1,ORDERN)
        ELSE
            WRITE(9,2541)KK,KREAL,(FTRAN(I),I=1,ORDERN)
        ENDIF
C
C*****************          CALCULATE   PSI(K,I,J)            ***********
C
        DO 950 I=1,ORDERN
            DO 950 J=1,ORDERN
  950   PSI(I,J) = PHI(I,J) + DEL(I,1) * FTRAN(J)
C
C*****************          CALCULATE   P (K,I,J)             ************
C
        DO 960 I=1,ORDERN
            DO 960 J=1,ORDERN
               DO 960 L=1,ORDERN
  960   GM(I,J) = GM(I,J) + PSI(L,I) * P(L,J)
        DO 980 I=1,ORDERN
           DO 980 J=1,ORDERN
              DO 970 L=1,ORDERN
  970       HM(I,J) = HM(I,J) + GM(I,L) * PSI(L,J)
                P(I,J) = HM(I,J) + Q(I,J) + R * FTRAN(I) * FTRAN(J)
  980   HM(I,J) = 0.0
        DO 990 I=1,ORDERN
           DO 990 J=1,ORDERN
  990   GM(I,J) = 0.0
C
C*****************  DISCRETE TIME VECTOR FOR PLOTTING GAINS  ************
C
        VTIME(KK) = KK
C
 1000 CONTINUE
C
C***************   DO YOU WANT TO SEE THE GAINS PLOTTED ?    ***********
C
 1001 WRITE(*,2545)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 1006
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 1002
        GOTO 1001
C
C*****************          LOOP TO PLOT OUT THE GAINS        ************
C
 1002 DO 1005 GAIN = 1,ORDERN
C
C*****************           SET THE GAIN PLOT TITLE          ************
C
           IF(GAIN.EQ.1)PNAME1 = 'FEEDBACK GAIN (F1) FOR STATE X1'
           IF(GAIN.EQ.2)PNAME1 = 'FEEDBACK GAIN (F2) FOR STATE X2'
           IF(GAIN.EQ.3)PNAME1 = 'FEEDBACK GAIN (F3) FOR STATE X3'
           IF(GAIN.EQ.4)PNAME1 = 'FEEDBACK GAIN (F4) FOR STATE X4'
           IF(GAIN.EQ.5)PNAME1 = 'FEEDBACK GAIN (F5) FOR STATE X5'
           IF(GAIN.EQ.6)PNAME1 = 'FEEDBACK GAIN (F6) FOR STATE X6'
           IF(GAIN.EQ.7)PNAME1 = 'FEEDBACK GAIN (F7) FOR STATE X7'
           IF(GAIN.EQ.8)PNAME1 = 'FEEDBACK GAIN (F8) FOR STATE X8'
```

```
            PNAM1L = 31.
C
C*****************************************************************
C****************              PLOT 88  GRAPHICS            ***********
C*****************************************************************
C
C****************  SET UP INITIAL PARAMETERS FOR GAIN PLOT  ***********
            BEGTIM    = 0.0
            FINTIM    = NSTAGE
            NPTS      = NSTAGE
            DO 1003 J = 1,7
              VYSS(J) = 0.0
 1003     VTIMSS(J) = ((FINTIM - BEGTIM)/6.)*(J-1)
C
C****************  GENERATE GAIN VECTOR FOR PLOTTING GAINS  ************
C
            DO 1004 KREAL = 1,NSTAGE
               KK         = NSTP1 - KREAL
               VY(KREAL) = FNEG(KK,GAIN)
C*********  TEST LINE FOR SELECTING PROPER COLUMN OF GAINS FOLLOWS ****
C*********              SEE LL51 FOR COMPILED VERSION              ****
C             WRITE(*,*) GAIN,KREAL,KK,FNEG(KK,GAIN)
 1004     CONTINUE
C
C****************             MAKE THE GAIN PLOT            ***********
C
            IF(GAIN .EQ. 1)PAUSE
            CALL GRAPH(99,99,1)
C
C**************** IS A HARDCOPY OF THE GAIN PLOT DESIRED ? ************
C
            WRITE(*,2595)
            READ (*,2190)ANSWER
              IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')
     +         CALL GRAPH(IOPORT,MODEL,1)
            CONTINUE
 1005 CONTINUE
C
C****************      DO YOU WANT TO CHANGE NSTAGE ?       ************
C
 1006 CHNGN = 0
       WRITE(*,2546)
      READ (*,2190)ANSWER
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')THEN
            CHNGN = 1
            GOTO 20
         ELSEIF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')THEN
            GOTO 1007
         ELSE
            GOTO1006
         ENDIF
C
C****************         IS A PHASE PLANE DESIRED ?        ************
C
 1007 WRITE(*,2547)
       READ (*,2190)ANSWER
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y') THEN
            NSTP1  = NSTAGE + 1
            PHASE  = 1
            GOTO 1025
         ENDIF
         IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 1010
         GOTO 1007
C
C****************        IS A TIME RESPONSE DESIRED ?       ************
C
 1010 PHASE = 0
       WRITE(*,2550)
       READ (*,2190)ANSWER
         IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 1020
```

```
          IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 1510
          GOTO 1010
C*****************          GRAPH IS TO BE A TIME RESPONSE      ************
 1020 NSTP1  = NSTAGE + 1
       PLTYPE = 3
C
C*****************            HOW MANY SECONDS  ?               ************
C
 1025 WRITE(*,2560)
       READ (*,*)TFINAL
C
C*****************         INPUT DT IF NOT ALREADY KNOWN        ************
C
       IF(DTFLAG .EQ. 0) THEN
           WRITE(*,2380)
           READ (*,*)DT
           DTFLAG = 1
       ENDIF
C
C*****************         CALCULATE FINAL VALUE OF K           ************
C
       KFINAL = NINT(TFINAL/DT)
       TFTEMP = KFINAL * DT
       IF(TFTEMP .LT. TFINAL) THEN
           KFINAL = KFINAL + 1
           TFINAL = KFINAL * DT
       ENDIF
C
C*****************    ENSURE THAT ENOUGH GAINS ARE CALCULATED   ************
C*****************        TO COVER THE DESIRED TIME RANGE       ************
C
       IF (GNSKED .EQ. 3) GOTO 1029
       IF((KFINAL-1) .GT. NSTAGE) THEN
           MAXTIM = DT * NSTAGE
           WRITE(*,2561)MAXTIM
           GOTO 1025
       ENDIF
C
C*****************      READ IN THE INITIAL STATE VECTOR        ************
C
 1029 WRITE(*,2565)
       DO 1030 I=1,ORDERN
           WRITE(*,2566)I
           READ (*,*)XK0(I,1)
 1030 CONTINUE
C
C*****************      READ IN THE COMMAND INPUT VECTOR        ************
C
       WRITE(*,2570)
       DO 1035 I=1,ORDERN
           WRITE(*,2580)I
           READ (*,*)INPUT(I,1)
 1035 CONTINUE
C
C*********      WRITE INITIAL STATE AND COMMAND INPUT VECTOR    **********
C*********                       TO  OUTPUT FILE                **********
C
       WRITE(9,2030)
       WRITE(9,2584)
       DO 1036 I = 1,ORDERN
           WRITE(9,2585) I,XK0(I,1),INPUT(I,1)
 1036 CONTINUE
C
C*****************      CHOOSE EITHER STEADY STATE GAINS (1)    ************
C*****************                OR  DYNAMIC GAINS      (2)    ************
C*****************              OR USER DEFINED GAINS    (3)    ************
C*****************      IF ONLY ONE CONTROL INPUT IS USED       ************
C
 1040 IF(NINPTS .EQ. 1) THEN
          IF(GAINCH .NE. 2)THEN
```

```
               WRITE(*,2590)
               READ (*,2070)TEMP
               CALL COMPARE(TEMP,1,3,CODE,IGOOD)
               IF(CODE.EQ.0)GOTO 1040
               GNSKED = IGOOD
            ELSE
               GAINCH = 1
            ENDIF
         ELSE
            GNSKED = 3
         ENDIF
         GOTO(141,142,143) GNSKED
C*****************          USE STEADY STATE GAINS             ************
 141   PNAME3 = 'OPTIMUM STEADY STATE GAIN SCHEDULE'
         PNAM3L = 34.
         GOTO 1054
C*****************           USE DYNAMIC GAINS                 ************
 142   PNAME3 = 'OPTIMUM DYNAMIC GAIN SCHEDULE'
         PNAM3L = 29.
         GOTO 1054
C*****************  IMPLEMENT  USER DEFINED FEEDBACK GAINS  ************
 143   PNAME2 = 'Implementing'
         PNAM2L = 12.
         PNAME3 = 'USER DEFINED GAINS'
         PNAM3L = 18.
         IF( GNSKD3 .EQ. 1 ) GOTO 1043
C      IF( FINAL .EQ. 1  .AND. GNSKD3 .EQ. 1 ) GOTO 1043
C*****************    INPUT USER DEFINED FEEDBACK GAINS     ************
 1044  DO 1045 I = 1,NINPTS
            DO 1045 J = 1,ORDERN
               WRITE(*,2592) I,J
               READ (*,*) USERGN(I,J)
 1045  CONTINUE
         GNSKD3 = 1
C***************** ECHO USER DEFINED FEEDBACK MATRIX         ************
C
 1043  WRITE(*,2593)
         DO 1046 I=1,NINPTS
 1046  WRITE(*,2594)(USERGN(I,J), J=1,ORDERN)
C
C*****************    MODIFY THE USER DEFINED GAINS IF NEEDED  *********
C
 1047  WRITE(*,2160)
         READ (*,2070)TEMP
         CALL COMPARE(TEMP,1,3,CODE,IGOOD)
         IF(CODE.EQ.0)GOTO 1047
         OPTION = IGOOD
         GOTO(1048,1044,1052)OPTION
           GOTO 1047
C
C********* CHANGE ONE ELEMENT OF USER DEFINED GAIN MATRIX  ************
C
 1048  WRITE(*,2170)
         READ (*,*)I,J
         IF(I.LT.1 .OR. I.GT.NINPTS .OR. J.LT.1
      +            .OR. J.GT.ORDERN)GOTO 1048
         WRITE(*,2592)I,J
         READ (*,*)USERGN(I,J)
         WRITE(*,2593)
         DO 1049 I=1,NINPTS
 1049  WRITE(*,2594)(USERGN(I,J),J=1,ORDERN)
 1051  WRITE(*,2180)
         READ (*,2190)ANSWER
           IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 1052
           IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 1048
           GOTO 1051
C
C*********   WRITE USER DEFINED GAIN VECTOR TO OUTPUT FILE   **********
C
 1052  CONTINUE
```

```
       WRITE(9,2030)
       WRITE(9,2593)
       DO 1053 I = 1,NINPTS
            WRITE(9,2594) (USERGN(I,J),J=1,ORDERN)
 1053 CONTINUE
       WRITE(9,2030)
 1054 IF(FINAL.EQ.1)GOTO 1520
       IF(PHASE .EQ. 0) GOTO 1050
C****************     CALCULATE STATES FOR PHASE PLANE       ************
       CALL STCALC(1,XK0,0,0)
C****************     SET UP INITIAL PARAMETERS FOR THE      ************
C****************              PHASE PLANE PLOT              ************
            NPTS      = KFINAL
C
C****************           PLOT THE PHASE PLANE             ************
C
            CALL GRAPH(99,99,2)
C
C****************     IS A HARDCOPY OF THE PLOT DESIRED ?    ************
C
            WRITE(*,2595)
            READ (*,2190)ANSWER
              IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')CALL GRAPH(IOPORT,MODEL,2)
            CONTINUE
            GOTO 1010
C
C****************   DO YOU WANT TO SEE THE TIME RESPONSE     ************
C****************           TABLE  ON THE SCREEN ?           ************
C
 1050 WRITE(*,2591)
       READ (*,2190)ANSWER
          IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')SCREEN = 0
          IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')SCREEN = 1
C
C****************   SELECT HOW THE STATES ARE TO BE PLOTTED ************
C
  151 WRITE(*,2598)
       READ (*,2070)TEMP
       CALL COMPARE(TEMP,1,3,CODE,IGOOD)
       IF(CODE.EQ.0)GOTO 151
       STPLOT = IGOOD
C
C****************     LOOP TO PLOT OUT STATE TRAJECTORIES    ************
C
       DO 1500 STVAR = 1,ORDERN
C
C****************     IS THIS STATE TO BE PLOTTED ?          ************
C
          IF(STPLOT .EQ. 2) THEN
             WRITE(*,2599) STVAR
             READ (*,2190)ANSWER
             IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')PLOTCH = 0
             IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')PLOTCH = 1
          ENDIF
C
C*************************************************************************
C****************      PRINT HEADING FOR OUTPUT TABLE        ************
C****************              TIME RESPONSE                 ************
C*************************************************************************
C
         IF(STVAR .EQ. 1) THEN
            IF(SCREEN .EQ. 1)THEN
               WRITE(*,2525)(HDG2(I),I=1,ORDERN)
               WRITE(*,2030)
            ENDIF
            WRITE(9,2525)(HDG2(I),I=1,ORDERN)
            WRITE(9,2030)
         ENDIF
C
C********             SKIP PLOTTING IF NO PLOT IS DESIRED     ************
```

```
C********    BUT MUST CALCULATE STATES ON FIRST TIME THROUGH ***********
C
            IF(STVAR .NE.1 ) THEN
               IF(STPLOT .EQ. 3) GOTO1499
               IF(STPLOT .EQ. 2  .AND.  PLOTCH .EQ. 0) GOTO1499
            ENDIF
C
C*****************     SET THE PLOT TITLE BASED ON THE      ************
C*****************              STATE SELECTED              ************
C
            IF(STVAR.EQ.1)PNAME1 = 'X1 TIME RESPONSE'
            IF(STVAR.EQ.2)PNAME1 = 'X2 TIME RESPONSE'
            IF(STVAR.EQ.3)PNAME1 = 'X3 TIME RESPONSE'
            IF(STVAR.EQ.4)PNAME1 = 'X4 TIME RESPONSE'
            IF(STVAR.EQ.5)PNAME1 = 'X5 TIME RESPONSE'
            IF(STVAR.EQ.6)PNAME1 = 'X6 TIME RESPONSE'
            IF(STVAR.EQ.7)PNAME1 = 'X7 TIME RESPONSE'
            IF(STVAR.EQ.8)PNAME1 = 'X8 TIME RESPONSE'
            PNAM1L = 16.
C
C*******************************************************************************
C*****************  CALL SUBROUTINE TO CALCULATE THE STATES  ************
C*******************************************************************************
C
          CALL STCALC(0,XK0,STVAR,SCREEN)
C
C*****************    SKIP PLOTTING IF NO PLOT IS DESIRED    ************
C
          IF(STPLOT .EQ. 3) GOTO1499
          IF(STPLOT .EQ. 2  .AND.  PLOTCH .EQ. 0) GOTO1499
C
C*******************************************************************************
C*****************              PLOT 88  GRAPHICS              ************
C*******************************************************************************
C
C*****************     SET UP INITIAL PARAMETERS FOR THE     ************
C*****************           STATE TRAJECTORY PLOT           ************
 1055     BEGTIM    = 0.0
            FINTIM    = TFINAL
            NPTS      = KFINAL
            DO 1060 J = 1,7
               VYSS(J)   = INPUT(STVAR,1)
               VTIMSS(J) = ((FINTIM - BEGTIM)/6.)*(J-1)
 1060     CONTINUE
C
C*****************        PLOT THE STATE TRAJECTORY          ************
C
            IF(STVAR .EQ. 1 ) PAUSE
            CALL GRAPH(99,99,3)
C
C*****************     IS A HARDCOPY OF THE PLOT DESIRED ?   ************
C
            WRITE(*,2595)
            READ (*,2190)ANSWER
              IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')CALL GRAPH(IOPORT,MODEL,3)
            CONTINUE
 1499 CONTINUE
 1500 CONTINUE
C
C**************   PRINT OUT THE AVERAGE VALUES OF ALL STATES ***********
C
       WRITE(*,2030)
       WRITE(*,2596)
       WRITE(9,2596)
       DO 1505 I=1,ORDERN
          WRITE(*,2597) I,AVG(I),I,AVG2(I),I,MAXVAL(I)
          WRITE(9,2597) I,AVG(I),I,AVG2(I),I,MAXVAL(I)
 1505 CONTINUE
       WRITE(*,2030)
       PAUSE
```

```
C
C****************     IS ANOTHER RUN OF OPTCON DESIRED ?    ************
C
      WRITE(9,2030)
 1510 WRITE(*,2600)
      READ (*,2190)ANSWER
        IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y')GOTO 1520
        IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n')GOTO 1530
        GOTO 1510
C
C****************           PRINT MENU OF OPTIONS           ************
C
 1520 WRITE(*,2610)
      READ (*,2070)TEMP
      CALL COMPARE(TEMP,1,11,CODE,IGOOD)
      IF(CODE.EQ.0)GOTO 1520
      OPTION = IGOOD
      IF(OPTION .LE. 4)THEN
           IF(NINPTS .GT. 1)THEN
              WRITE(*,2620)
              READ(*,2070)TEMP
              GOTO 1520
           ENDIF
      ENDIF
      IF(OPTION .EQ.2  .OR. OPTION .EQ.3) LOOP = 1
      IF(OPTION .EQ.8) GAINCH =1
      IF(OPTION .EQ.10  .AND. GAINCH .EQ.1) GAINCH =2
      FINAL  = 1
      GOTO(20,230,100,310,390,560,620,1040,10,780,1510)OPTION
        GOTO 1520
C
C***********
 1530 STOP
C***********
c
C
C*************************************************************************
C****************             FORMAT STATEMENTS             ************
C*************************************************************************
C
 2000 FORMAT(/,5X,'OPTCON minimizes the following cost',
     +' function:',//,5X,'J = MIN  ( X''(N) * H * X(N) + ',
     +' Sum( X''(k) * Q * X(k) + U''(k)',
     +' * R * U(k)))',//,5X,'The output of the program is the',
     +' feedback gain matrix, F transpose, (F''),',/,5X,'which, when',
     +' multiplied by the State Vector (X),',/,5X,'yields a scalar',
     +' control.(U).',///,5X,'The following recursive equations ',
     +'were derived using dynamic programming,',/,5X,
     +'starting at the terminal time.(N) and working backwards:',//)
 2010 FORMAT(8X,
     +'(1) F''(k)  = -(DEL''*P(k-1)*PHI)/(DEL''*P(k-1)*DEL + R)',3X,
     +'F''(0)=0',/,8X,
     +'(2) PSI(k) = PHI + DEL*F''(k)',27X,'PSI(0)=0',/,8X,
     +'(3) P(k)   = PSI''(k)*P(k-1)*PSI(k) + Q + F''(k)*R*F(k)',4X,
     +'P(0)=H',////)
 2015 FORMAT(/,5X,'You may enter a system with either single or',
     +' multiple control signals. ',/,9X,' If a system with only one',
     +' control signal is entered,',/,9x,' then the optimal gains can',
     +' be generated as described',/,9x,' above. These gains may then',
     +' be implemented into the',/,9x,' state equations to obtain a',
     +' time response of the system.',/,9x,' If you choose to enter a',
     +' system with multiple control signals,',/,9x,' then you must',
     +' enter the feedback gains manually.  The user defined',/,9X,
     +' gains option exists for the single control input system also.',
     +,////,1X,'First enter the problem',
     +1X,'identification ( NOT to exceed 20 characters ).',//,
     +10X,'PROBLEM ID........', )
 2020 FORMAT(A20)
 2030 FORMAT(' ',/,70('*'),/)
 2040 FORMAT(///,5X,'OPTIMAL CONTROL PROGRAM',/)
```

```
2050 FORMAT(6X,/,' PROBLEM IDENTIFICATION:',5X,A20, )
2055 FORMAT(5X,//,' Select the type of printer that you are',
     +' using ',/,
     +'                ( Answer    1 or 2 ) ',//,
     +10X,'1)  EPSON or THINKJET',/
     +10X,'2)  LASERJET',//,
     +10X,'ANSWER.....', )
2060 FORMAT(5X,//,' Enter the ORDER of the system (up to 8).  ', )
2070 FORMAT(A2)
2075 FORMAT(5X,//,' Enter the NUMBER OF CONTROL INPUTS (up to 8).',//,
     +   5X,' NOTE...NO OPTIMAL GAINS will be generated if you enter',/,
     +   5X,'        more than one control input',//,
     +   5x,' ANSWER.......?  ', )
2076 FORMAT(/,5X,' The NUMBER OF CONTROL INPUTS  = ',I1)
2077 FORMAT(/,5X,' Any changes to NUMBER OF CONTROL INPUTS ?   '
     +          '  (Answer  y or n)  ', )
2080 FORMAT(5X,//,' Enter the NUMBER of TIME INTERVALS (N) over',
     +         ' which the cost function',/,' is to be',
     +         ' minimized.  (MUST NOT exceed 1000)   ', )
2090 FORMAT(10X,//,' Does the cost function (J) include the State',
     +' TRAJECTORY over all stages ?',/,
     +'                ( Answer    1,2,or 3 ) ',//,
     +10X,'1)  YES...Set  Q  equal to the IDENTITY Matrix .',/
     +10X,'2)  YES...Each diagonal element of  Q  will be entered'
     +' separately .',/
     +10X,'3)  NO....Set  Q  equal to the   ZERO   Matrix .',//,
     +10X,'ANSWER.....', )
2100 FORMAT(9X,//,' The states are weighted equally for the',
     +' TRAJECTORY over all stages.')
2110 FORMAT(9X,//,' Enter the elements of the Q matrix.',/,
     +' (State weighting matrix for TRAJECTORY over all stages)',/)
2120 FORMAT(9X,//,' The state TRAJECTORY is not included in your',
     +' cost function.')
2130 FORMAT(6X,'Q(',I1,',',I1,') = ', )
2140 FORMAT(//,5X,' The Q Matrix ',/)
2150 FORMAT(2X,8(F8.3,1X))
2160 FORMAT(//,5X,'Do you want to change any element of the matrix?',
     +//,10X,'1)  YES...a SINGLE element.',/
     +10X,'2)  YES...the ENTIRE Matrix.',/
     +10X,'3)  NO',//,
     +10X,'ANSWER.....', )
2170 FORMAT(/,5X,' Which element of the Matrix do you want to',
     +' Change ?',/,
     +5X,' If I is the ROW and J is the COLUMN,....enter   I,J   ', )
2180 FORMAT(10X,//,5X,' Any other changes?  (Answer  y or n)  ', )
2190 FORMAT(A1)
2200 FORMAT(10X,//,' Does the cost function (J) include TERMINAL',
     +' States ?  ( Answer    1,2,or 3 ) ',//,
     +10X,'1)  YES...Set  H  equal to the IDENTITY Matrix .',/
     +10X,'2)  YES...Each diagonal element of  H  will be entered'
     +' separately .',/
     +10X,'3)  NO....Set  H  equal to the   ZERO   Matrix .',//,
     +10X,'ANSWER.....', )
2210 FORMAT(9X,//,' All states are weighted equally for the',
     +' TERMINAL states.')
2220 FORMAT(9X,//,' Enter the elements of the H matrix.',/,
     +' (State weighting matrix for TERMINAL states)',/)
2230 FORMAT(9X,//,' The TERMINAL states are not included in your',
     +' cost function.')
2240 FORMAT(6X,'H(',I1,',',I1,') = ', )
2250 FORMAT(//,5X,' The H Matrix ',/)
2260 FORMAT(//,5X,' Enter the value of the scalar R',/,
     +5X,' (Control input weighting factor)',//,5X,' R = ?  ', )
2270 FORMAT(/,5X,' The scalar R = ',F8.4)
2280 FORMAT(/,5X,' Any changes to R ?  (Answer  y or n)  ', )
2290 FORMAT(///,4X,
     +' If you want to read in the A and B matrices for a CONTINUOUS',
     +'  TIME system ,',/,4X,
     +'.....Enter 0',
     +//,4X,' If you want to enter the PHI and DEL matrices for a',
```

```
     +'  DISCRETE TIME system,',/,4X,
     +' ..................................................Enter 1',//,
     +10X,'ANSWER.....',  )
2300 FORMAT(/,5X,' You will enter the A and B matrices. ',/,
     +5X,'  .......Is this correct ?  ', )
2310 FORMAT(/,5X,' You will enter the PHI and DEL matrices. ',/,
     +5X,'  .........Is this correct ?  ', )
2320 FORMAT(5X,//,' Enter the elements of the plant matrix...A.',/)
2330 FORMAT(6X,'A(',I1,',',I1,') = ', )
2335 FORMAT(5X,/,' No changes to  A  or  B  will be allowed because',/,
     +       5X,' you have entered a DISCRETE TIME system',/,
     +       5X,' Hit    ENTER    to continue.....', )
2340 FORMAT(//,5X,' The A Matrix (Plant Matrix)',/)
2350 FORMAT(5X,/' Enter the elements of the control distribution',
     +            ' matrix...B.',/)
2360 FORMAT(6X,'B(',I1,',',I1,') = ', )
2370 FORMAT(//,5X,' The B Matrix (Control Distribution Matrix)',/)
2380 FORMAT(5X,/,' Enter the SAMPLE INTERVAL.....DT = ?  ', )
2385 FORMAT(5X,/,' No changes to DT will be allowed because',
     +       ' you have entered a DISCRETE TIME system',/,
     +       5X,' Hit    ENTER    to continue.....', )
2390 FORMAT(//,5X,' The SAMPLE INTERVAL DT = ',F8.4)
2400 FORMAT(/,5X,' Any changes to the SAMPLE INTERVAL ?  (Answer',
     +'  y or n)  ', )
2410 FORMAT(5X,//,' Enter the elements of the PHI matrix.',/)
2420 FORMAT(6X,'PHI(',I1,',',I1,') = ', )
2425 FORMAT(5X,/,' No changes to PHI or DEL will be allowed because',/,
     +       5X,' you have entered a CONTINUOUS TIME system',/,
     +       5X,' Hit    ENTER    to continue.....', )
2430 FORMAT(//,5X,' The PHI Matrix',/)
2440 FORMAT(5X,//,' Enter the elements of the DEL matrix.',/)
2450 FORMAT(6X,'DEL(',I1,',',I1,') = ', )
2460 FORMAT(//,5X,' The DEL Matrix',/)
2470 FORMAT(///,5X,' The ORDER of the system = ',I1)
2475 FORMAT(///,5X,' The NUMBER OF CONTROL INPUTS = ',I1)
2480 FORMAT(///,5X,' The NUMBER of TIME INTERVALS = ',I3,/)
2490 FORMAT(2X,8(F8.3,1X))
2500 FORMAT(//,' Minimum TERMINAL STATES Control')
2510 FORMAT(//,' Minimization over ALL STAGES')
2515 FORMAT(//,4X,' Do you want to see the gain schedule table on',
     +' the screen ?',//,5X,'(Answer  y or n) ', )
2520 FORMAT(//,'  NEG',',    REAL',/,
     +          '  TIME',',    TIME',T18,4(A5,5X),/,
     +          '  STEP',',    INDEX',T18,4(A5,5X),//)
2525 FORMAT(//,'  REAL',/,
     +          '  TIME    REAL',T20,4(A4,8X),/,
     +          '  INDEX   TIME',T20,4(A4,8X),//)
2530 FORMAT(/,' Optimum gains are reached after ',I3,'  stages.',
     +/,' The program is terminated early in order to',
     +' prevent a division by zero.',/)
2540 FORMAT(' ',2(I4,2X),T16,4(F8.4,2X),/,T16,4(F8.4,2X))
2541 FORMAT(' ',2(I4,2X),T16,4(F8.4,2X))
2545 FORMAT(//,4X,' Do you want to see the gains plotted ?',
     +//,5X,'(Answer  y or n) ', )
2546 FORMAT(//,4X,' Do you want to change the NUMBER OF STAGES ?',
     +//,5X,'(Answer  y or n) ', )
2547 FORMAT(//,4X,' Do you want to see a PHASE PLANE of X1 .vs.',
     +' X2 ?',//,5X,'(Answer  y or n) ', )
2550 FORMAT(//,4X,' Do you want to see a time response of your',
     +' system ?',//,5X,'(Answer  y or n) ', )
2560 FORMAT(//,4X,' For how many seconds ? ', )
2561 FORMAT(5X,//' The optimal gains are computed for only ',F8.4 ,
     +          ' seconds.',/,' Please enter a smaller number.')
2565 FORMAT(5X,/' Enter the elements of the INITIAL STATE vector ',
     +          ' - Xk(0)',/)
2566 FORMAT(6X,'X',I1,'(0) = ', )
2570 FORMAT(5X,/' Enter the elements of the COMMAND INPUT vector-R.',/)
2580 FORMAT(6X,'R(',I1,') = ', )
2584 FORMAT(T5,'  N   ',T14,'INITIAL STATE',T35,'COMMAND INPUT')
2585 FORMAT(T7,I1,T16,F9.4,T37,F9.4)
```

```
 2590 FORMAT(10X,//,' Select a gain schedule...( Answer  1,2,or 3 )',//,
     +10X,'1)  Use STEADY STATE OPTIMAL gains over all steps .',/
     +10X,'2)  Use DYNAMIC gains .',/
     +10X,'3)  Use STEADY STATE USER DEFINED gains .',//,
     +10X,'ANSWER.....', )
 2591 FORMAT(//,4X,' Do you want to see the time response table on',
     +' the screen ?',//,5X,'(Answer  y or n) ', )
 2592 FORMAT(6X,/,' CONTROL GAIN  F(',I1,',',I1,') = ?  ', )
 2593 FORMAT(//,5X,' The USER DEFINED GAIN Matrix',/)
 2594 FORMAT(' ',8(F7.4,1X))
 2595 FORMAT(//,4X,' Do you want a hardcopy of this plot ? ',
     +       '  ( Answer  y or n )  ', )
 2596 FORMAT(6X,' ',//,8X,' AVERAGE AND MAX VALUES OF ALL STATES',//)
 2597 FORMAT(6X,' Average Value of X',I1,'       =  ',E12.4,/,
     +       6X,' Average Value of X',I1,'     2    =  ',E12.4,/,
     +       6X,' Maximum Value of X',I1,'       =  ',E12.4,//)
 2598 FORMAT(//,5X,'Do you want to PLOT.....',
     +//,10X,'1)  ALL state trajectories.',/
     +10X,'2)  Only SELECTED state trajectories.',/
     +10X,'3)  NO state trajectories.',//,
     +10X,'ANSWER.....', )
 2599 FORMAT(//,5X,'Do you want to see a PLOT for state X',I1,' ? ',/,
     +           18X,'  (Answer  y or n)  ', )
 2600 FORMAT(//,4X,' This concludes the optimal control program',
     +' (OPTCON).',//,5X,'Do you want to run the program',
     +' again?  (Answer  y or n)  ', )
 2610 FORMAT(///,5X,' SELECT ONE OF THE FOLLOWING OPTIONS:',/,
     +10X,/,'  1) Change the NUMBER of STAGES.....N',/,
     +10X,/,'  2) Change the TERMINAL state weighting matrix.....H',/,
     +10X,/,'  3) Change the TRAJECTORY state weighting matrix...Q',/,
     +10X,/,'  4) Change the CONTROL weighting factor.....R',/,
     +10X,/,'  5) Change the present A and B matrices',/,
     +10X,/,'  6) Change the SAMPLE INTERVAL.....DT',/,
     +10X,/,'  7) Change the present PHI and DEL matrices',/,
     +10X,/,'  8) Change (or select) different FEEDBACK GAINS',/,
     +10X,/,'  9) Input an entirely NEW SYSTEM',/,
     +10X,/,' 10) NO CHANGES...RUN',/,
     +10X,/,' 11) EXIT the program',//,
     +10X,'SELECTION...( MUST be a number between 1 and 11 ).....', )
 2620 FORMAT(5X,/,' No change to  this parameter is allowed because',
     +       ' you have entered a MIMO system.',/,
     +      5X,' Hit    ENTER    to continue.....', )
C    - --
       END
```

APPENDIX C
OPTCON SUBROUTINE LISTINGS

The following subroutines are written in MICROSOFT Fortran and are to be used on an IBM compatible system. These subroutines are required by the main OPTCON program found in Appendix B and by the PLOT88 subroutine found in Appendix D. A brief synopsis of the subroutine functions is given below.

PHIDEL	-	Convert the continuous time **A** and **B** system matrices to the corresponding discrete time Φ and Γ matrices.
PROD	-	Perform simple matrix multiplication of two matrices. Maximum dimension of the matrices is limited to eight.
SUM	-	Perform simple matrix addition or subtraction of two matrices. Maximum dimension of the matrices is limited to eight.
COMPARE	-	Test a user input response to determine if the response lies within the range of allowable integers.
CLRSCR	-	A DOS command that allows the monitor screen to be cleared prior to the generation of a new graph.
GOTOXY	-	A DOS command that positions the cursor to a designated coordinate position on the monitor screen.
STCALC	-	Calculates the time response of system by iterating the discrete state equations.

```
$NOdebug
C                                LL63sub            12JULY87
C                                                   OK  SDL
C                                                   NEW output format
C                                                       for states
C*************************************************************************
C****************              SUBROUTINES                    ************
C*************************************************************************
C
C
C
      SUBROUTINE PHIDEL(T,ORDERN,M)
C
      COMMON /BLK1/ A,B,PHI,DEL
      INTEGER*2     ORDERN,I,J,ERFLAG
      REAL*4        A(8,8),B(8,2),PHI(8,8),DEL(8,2),
     +              PSIT(8,8),TERM(8,8),NEXTRM(8,8),ARATIO(8,8),
     +              TRATIO,ERROR,K
C
      ERROR  = 1.E-7
      ERFLAG = 0
      TRATIO = T*T/2.
C
      DO 1 I = 1,ORDERN
         DO 1 J = 1,ORDERN
            TERM(I,J)= A(I,J) * TRATIO
            IF(I .EQ. J)THEN
                PSIT(I,I) = T + TERM(I,I)
```

```
            ELSE
                PSIT(I,J) = TERM(I,J)
            ENDIF
      1 CONTINUE
C
C
         K = 2.
C
      2 K = K + 1.
         TRATIO = T/K
         DO 3 I = 1,ORDERN
            DO 3 J = 1,ORDERN
               ARATIO(I,J)= A(I,J) * TRATIO
      3 CONTINUE
C
         CALL PROD(TERM,ARATIO,ORDERN,ORDERN,ORDERN,NEXTRM)
C
         DO 4 I = 1,ORDERN
            DO 4 J = 1,ORDERN
               IF(ABS(NEXTRM(I,J)) .GE. ERROR) THEN
                   ERFLAG = ERFLAG + 1
               ENDIF
               TERM(I,J) = NEXTRM(I,J)
      4 CONTINUE
C
C
         IF(ERFLAG .GT. 0)THEN
             DO 5 I = 1,ORDERN
                DO 5 J = 1,ORDERN
                   PSIT(I,J)= TERM(I,J) + PSIT(I,J)
      5     CONTINUE
C
             ERFLAG = 0
             GOTO 2
         ENDIF
C*****************  NOTE THE DUAL USE OF 'TERM' HERE  **************
         CALL PROD(A,PSIT,ORDERN,ORDERN,ORDERN,TERM)
         DO 6 I = 1,ORDERN
            DO 6 J = 1,ORDERN
               IF(I .EQ. J)THEN
                   PHI(I,I) = 1.+ TERM(I,I)
               ELSE
                   PHI(I,J) = TERM(I,J)
               ENDIF
      6 CONTINUE
         CALL PROD(PSIT,B,ORDERN,ORDERN,M,DEL)
C
         RETURN
         END
C
C
C
C*******************************************************************************
C
         SUBROUTINE SUM(M1,M2,OPER,N,M,MSUM)
C
         INTEGER*2  N,M,OPER,I,J
         REAL*4     M1(8,8),M2(8,8),MSUM(8,8)
C
         DO 1 I=1,N
             DO 1 J=1,M
      1 MSUM(I,J)=0.0
C
C*****************  DO YOU WANT TO ADD OR SUBTRACT ?  **************
C
         IF(OPER) 2,2,3
C*****************                SUBTRACT                **************
      2 DO 20 I = 1,N
            DO 20 J = 1,M
                  MSUM(I,J) = M1(I,J) - M2(I,J)
```

```
    20 CONTINUE
       GOTO 40
C*****************              ADD                    **************
     3 DO 30 I = 1,N
          DO 30 J = 1,M
               MSUM(I,J) = M1(I,J) + M2(I,J)
    30 CONTINUE
    40 RETURN
       END
C
C
C
C*************************************************************************
C
       SUBROUTINE PROD (M1,M2,ORDERN,M,L,MPROD)
C
       INTEGER*2  ORDERN,M,L,I,J,K
       REAL*4     M1(8,8),M2(8,8),MPROD(8,8)
       DO 1 I=1,ORDERN
          DO 1 J=1,L
     1 MPROD(I,J)=0.0
       DO 2 I=1,ORDERN
          DO 2 J=1,L
             DO 2 K = 1,M
     2 MPROD(I,J) = MPROD(I,J) + M1(I,K) * M2(K,J)
       RETURN
       END
C
C
C
C*************************************************************************
C
       SUBROUTINE COMPARE (TEMP,VALMIN,VALMAX,CODE,IGOOD)
C
       INTEGER*2   IGOOD,CODE,VALMAX,VALMIN
       CHARACTER*2 TEMP
C
            IGOOD = -1
            IF(TEMP.EQ.'0')IGOOD=0
            IF(TEMP.EQ.'1')IGOOD=1
            IF(TEMP.EQ.'2')IGOOD=2
            IF(TEMP.EQ.'3')IGOOD=3
            IF(TEMP.EQ.'4')IGOOD=4
            IF(TEMP.EQ.'5')IGOOD=5
            IF(TEMP.EQ.'6')IGOOD=6
            IF(TEMP.EQ.'7')IGOOD=7
            IF(TEMP.EQ.'8')IGOOD=8
            IF(TEMP.EQ.'9')IGOOD=9
            IF(TEMP.EQ.'10')IGOOD=10
            IF(TEMP.EQ.'11')IGOOD=11
C
            IF(IGOOD.EQ.-1 .OR. IGOOD.GT.VALMAX .OR. IGOOD.LT.VALMIN) THEN
               CODE = 0
            ELSE
               CODE = 1
            ENDIF
       RETURN
       END
C
C
C
C*************************************************************************
C
       SUBROUTINE CLRSCR
C
       INTEGER*2   IC(4)
       CHARACTER*1 C1,C2,C3,C4
       EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
       DATA        IC/16#1B,16#5B,16#32,16#4A/
C
```

```
C *** Write Escape Code to Display ***
      WRITE(*,1) C1,C2,C3,C4
    1 FORMAT(1X,4A1)
C
      RETURN
      END
C
C
C
C*********************************************************************
C
      SUBROUTINE GOTOXY(ROW,COLUMN)
C
C     ***** Position Cursor by Row,Column *****
C
      INTEGER*2 IC(4),ROW,COLUMN,L
      CHARACTER*1 C1,C2,C5,C8,LC(5)
      CHARACTER*5 CBUFF
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),
     +            (CBUFF,LC(1))
      DATA        IC/16#1B,16#5B,16#3B,16#66/
C
      L=10000+100*ROW+COLUMN
C
C *** Write Escape Codes to a Character Buffer ***
      WRITE(CBUFF,2) L
    2 FORMAT(I5)
C
C *** Write Escape Codes to Display ***
      WRITE(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
    3 FORMAT(1X,8A1, )
      RETURN
      END
C
C
C
C*********************************************************************
C
      SUBROUTINE STCALC(PHASE,XK0,STVAR,SCREEN)
C
C *** CALCULATE THE STATES ITERATIVELY  ***
C ***  X(k+1) = PHI * X(k) + DEL * U(k) ***
C
      COMMON /BLK1/ A,B,PHI,DEL
      COMMON /BLK3/ VTIME,VTIMSS,VY,VYSS,VXXSS,VXYSS
      COMMON /BLK4/ KFINAL,NSTAGE,NSTP1,ORDERN,GNSKED,USERGN,FNEG,
     +              INPUT,DT,AVG,AVG2,MAXVAL,NINPTS
      INTEGER*2     KFINAL,NSTAGE,NSTP1,ORDERN,GNSKED,STVAR,SCREEN,
     +              PHASE,M,OPER,NINPTS,NINPP1
      REAL*4        A(8,8),B(8,2),PHI(8,8),DEL(8,2),VTIME(1002),
     +              VTIMSS(9),VY(1002),VYSS(9),FNEG(1000,8),INPUT(8,1),
     +              DT,PHIEQX(8,1),PHIEQ(8,8),DELROW(8,8),ROWF(8,8),
     +              XK0(8,1),XK(8,1),XKP1(8,1),VXXSS(9),VXYSS(9),
     +              DELINP(8,1),AVG(8),AVG2(8),STSUM,STSUM2,MAXVAL(8),
     +              USERGN(8,8)
C
C****************       RE-INITIALIZE THE STATE VECTOR      ************
C
      DO 5 J = 1,ORDERN
         XK(J,1)   = XK0(J,1)
    5 CONTINUE
C
C****************       RE-INITIALIZE THE AVERAGING SUMS    ************
C****************       AND MAXIMUM VALUE STATE VECTOR      ************
C
      STSUM         = 0.0
      STSUM2        = 0.0
      MAXVAL(STVAR) = 0.0
C
C****************  LOOP TO ITERATIVELY CALCULATE THE STATES ************
```

```
C
      DO 70 K = 1,KFINAL
           KPRIME   = NSTP1 - K
           TIME     = K * DT
           IF (PHASE .EQ. 1) THEN
               VY(K)    = XK(2,1)
               VTIME(K) = XK(1,1)
           ELSE
               VY(K)    = XK(STVAR,1)
               VTIME(K) = TIME
C**************** SUM FOR COMPUTING AVERAGE STATE VALUES  ************
               STSUM  = STSUM  + XK(STVAR,1)
               STSUM2 = STSUM2 + (XK(STVAR,1) * XK(STVAR,1))
C****************    SEARCH FOR MAXIMUM VALUE OF THE STATE  ************
               IF( ABS( XK(STVAR,1) )  .GT.  ABS( MAXVAL(STVAR) ))
     +           MAXVAL(STVAR) = XK(STVAR,1)
           ENDIF
C
C**************** LOOP TO SELECT THE PROPER FEEDBACK GAIN  ************
C****************      ELEMENTS FOR THIS TIME STEP          ************
C
           DO 40 J = 1,ORDERN
              GOTO(10,20,35)GNSKED
C****************          USE STEADY STATE GAINS  (GNSKED=1)************
   10        ROWF(1,J) = FNEG(NSTAGE,J)
              GOTO 30
C****************           USE DYNAMIC GAINS      (GNSKED=2)************
   20         IF(K .LE. NSTAGE) THEN
                 ROWF(1,J) = FNEG(KPRIME,J)
              ELSE
                 ROWF(1,J) = 0.0
              ENDIF
   30         CONTINUE
C****************       USER DEFINED GAINS         (GNSKED=3)************
   35         CONTINUE
   40      CONTINUE
C
C****************       PAD THE DEL AND ROWF MATRICES       ***********
C****************                 WITH ZEROS                ***********
C****************   IN ORDER TO MULTIPLY PROPERLY IN PROD   ***********
C
           NINPP1  = NINPTS + 1
           DO 50 I = 1,ORDERN
              DO 50 J = NINPP1,ORDERN
                 DEL(I,J)    = 0.0
                 ROWF(J,I)   = 0.0
   50      CONTINUE
C
C*************************************************************************
C****************      CALCULATE THE NEXT STATE  X(k+1)     ************
C*************************************************************************
C
           M = 1
           IF(GNSKED .NE. 3) THEN
C****************       USING OPTIMAL GAIN SCHEDULE          ************
              CALL PROD(DEL,ROWF,ORDERN,ORDERN,ORDERN,DELROW)
           ELSE
C****************       USING USER DEFINED GAIN MATRIX       ************
              CALL PROD(DEL,USERGN,ORDERN,ORDERN,ORDERN,DELROW)
           ENDIF
           OPER = 1
           CALL SUM(PHI,DELROW,OPER,ORDERN,ORDERN,PHIEQ)
           CALL PROD(PHIEQ,XK,ORDERN,ORDERN,M,PHIEQX)
           CALL PROD(DELROW,INPUT,ORDERN,ORDERN,M,DELINP)
           OPER = 0
           CALL SUM(PHIEQX,DELINP,OPER,ORDERN,M,XKP1)
C*************************************************************************
C
C****************  NEXT 29 LINES ARE TEST LINES TO VERIFY   ************
C****************     PROPER CALCULATION OF THE STATES      ************
```

```
C****************     FOR A SECOND ORDER OPTIMAL EXAMPLE     ************
C
C         WRITE(*,2614) K
C         WRITE(*,2615)
C         DO 1041 I = 1,ORDERN
C             WRITE(*,2620)DEL(I,1),ROWF(1,I),(DELROW(I,J),J=1,ORDERN)
C1041     CONTINUE
C         WRITE(*,2625)
C         DO 1042 I = 1,ORDERN
C             WRITE(*,2630)(PHI(I,J),J=1,ORDERN),(DELROW(I,J),
C    +                      J=1,ORDERN),(PHIEQ(I,J),J=1,ORDERN)
C1042     CONTINUE
C         WRITE(*,2635)
C         DO 1043 I = 1,ORDERN
C             WRITE(*,2640)(PHIEQ(I,J),J=1,ORDERN),XK(I,1),PHIEQX(I,1)
C1043     CONTINUE
C         WRITE(*,2645)
C         DO 1044 I = 1,ORDERN
C             WRITE(*,2650)PHIEQX(I,1),DELINP(I,1),XKP1(I,1)
C1044     CONTINUE
C2614 FORMAT(/,'       TIME STEP = ',I3,/)
C2615 FORMAT(/,T10,' DEL',T22,'ROWF TRAN',T44,'DELROW')
C2620 FORMAT(T5,F10.4,T20,F10.4,T35,2(F10.4))
C2625 FORMAT(/,T10,' PHI',T38,'  DELROW ',T66,'PHIEQ ')
C2630 FORMAT(2(F10.4),8X,2(F10.4),8X,2(F10.4))
C2635 FORMAT(/,T10,' PHIEQ',T33,'     XK    ',T51,'PHIEQX')
C2640 FORMAT(2(F10.4),8X,F10.4,8X,F10.4)
C2645 FORMAT(/,T10,' PHIEQX',10X,' DELINP ',12X,' XKP1 ')
C2650 FORMAT(T5,3(F10.4,8X))
C*****************************************************************
C************      NEXT 24 LINES ARE TEST LINES TO VERIFY      ********
C************         PROPER CALCULATION OF THE STATES         ********
C************   FOR A FOURTH ORDER USER DEFINED GAINS EXAMPLE  ********
C*****************************************************************
C
C         WRITE(9,2614) K
C         WRITE(9,2615)
C         DO 1041 I = 1,ORDERN
C             WRITE(9,2620)(PHI(I,J),J=1,ORDERN),(DEL(I,J),J=1,NINPTS)
C1041     CONTINUE
C         WRITE(9,2625)
C         DO 1042 I = 1,ORDERN
C             WRITE(9,2630)(DELROW(I,J),J=1,ORDERN),DELINP(I,1),
C    +                      (USERGN(J,I),J=1,NINPTS)
C1042     CONTINUE
C         WRITE(9,2635)
C         DO 1043 I = 1,ORDERN
C             WRITE(9,2640)(PHIEQ(I,J),J=1,ORDERN),PHIEQX(I,1),
C    +                     XKP1(I,1),XK(I,1)
C1043     CONTINUE
C2614 FORMAT(/,'       TIME STEP = ',I3,/)
C2615 FORMAT(/,T10,' PHI',T57,'DEL')
C2620 FORMAT(T5,4(F7.4,2X),T50,2(F7.4,2X))
C2625 FORMAT(/,T10,' DELROW',T52,'DELINP ',T67,'USERGN')
C2630 FORMAT(T5,4(F7.4,2X),T50,F7.4,T62,2(F7.4,2X))
C2635 FORMAT(/,T10,' PHIEQ',T51,'PHIEQX',T63,'XKP1',T74,'XK')
C2640 FORMAT(T5,4(F7.4,2X),T50,F7.4,T60,F7.4,T70,F7.4)
C
C****************         PRINT OUT THE STATE TABLE         ************
C****************              ONLY ONCE                    ************
C
           IF(PHASE .NE. 1) THEN
              IF(STVAR .EQ. 1) THEN
                 IF(SCREEN .EQ. 1.) THEN
                    IF(ORDERN .GT. 4 ) THEN
                       WRITE(*,2670)K,TIME,(XK(I,1),I=1,ORDERN)
                    ELSE
                       WRITE(*,2671)K,TIME,(XK(I,1),I=1,ORDERN)
                    ENDIF
                 ENDIF
```

```
                    IF(ORDERN .GT. 4 ) THEN
                       WRITE(9,2670)K,TIME,(XK(I,1),I=1,ORDERN)
                    ELSE
                       WRITE(9,2671)K,TIME,(XK(I,1),I=1,ORDERN)
                    ENDIF
 2670          FORMAT('  ',I4,T7,F8.4,T15,4(F10.4,2X),/,T15,4(F10.4,2X))
 2671          FORMAT('  ',I4,T7,F8.4,T15,4(F10.4,2X))
               ENDIF
            ENDIF
C
C*****************     GET READY FOR THE NEXT ITERATION      ************
C
            DO 60 I = 1,ORDERN
                XK(I,1) = XKP1(I,1)
   60     CONTINUE
   70 CONTINUE
C
C*****************     CALCULATE THE AVERAGE OF THE STATE    ************
C*****************       BEING CONSIDERED ON THIS CALL       ************
C
       IF(STVAR .NE. 0)THEN
          AVG(STVAR)  = STSUM/KFINAL
          AVG2(STVAR) = STSUM2/KFINAL
       ENDIF
       RETURN
       END
```

APPENDIX D
PLOT88 GRAPHICS SUBROUTINE LISTING

The following code is written in MICROSOFT Fortran and is intended to be used on an IBM compatible system. This graphics subroutine must be linked with the two program segments found in Appendices B and C. In addition, the Fortran, Math and PLOT88 libraries must be linked.

```
$NOdebug
C                                LL63GR                    OK  SDL
C                                                          12 JULY 87
C*************************************************************************
C****************              SUBROUTINES                  ************
C*************************************************************************
C
C
C
      SUBROUTINE GRAPH(IOPORT,MODEL,PLTYPE)
C
      IMPLICIT REAL*4 (A-Z)
      COMMON /BLK2/ BEGTIM,FINTIM,NPTS,
     +              XNAML,YNAML,PNAM1L,PNAM2L,PNAM3L
      COMMON /BLK3/ VTIME,VTIMSS,VY,VYSS,VXXSS,VXYSS
      COMMON /BLK5/ XNAME,YNAME,PNAME1,PNAME2,PNAME3
      INTEGER*2     NPTS,IOPORT,MODEL,XNAML,YNAML,
     +              NCHAR1,NCHAR2,NCHAR3,PLTYPE,J
      REAL*4        VTIME(1002),VY(1002),XAXL,YAXL,VTIMSS(9),VYSS(9),
     +              XORGN,YORGN,VXXSS(9),VXYSS(9),
     +              XLO,XHI,YLO,YHI,INCRMT
      CHARACTER*30  XNAME,YNAME
      CHARACTER*51  PNAME1,PNAME2
C
      IF(MODEL .EQ. 99)THEN
C*****************          SEND TO MONITOR           **************
          CALL CLRSCR
          XORGN = 1.50
          YORGN = 0.80
      ELSE
C*****************          SEND TO PLOTTER           **************
          XORGN = 3.20
          YORGN = 1.76
      ENDIF
C
   10 CALL GOTOXY(10,25)
      WRITE(*,*) 'Calculating Plotting Data'
C
      IF (PLTYPE .EQ. 1) THEN
C*****************        PLOTTING THE GAINS          **************
          XAXL   = 5.0
          XOFF   = 0.25
          XNAME  = 'DISCRETE REAL TIME INDEX (k)'
          XNAML  = -28
          YNAME  = 'GAIN TRAJECTORY'
          YNAML  =  15
          PNAME3 = ' '
          PNAM3L = 1
      ELSEIF (PLTYPE .EQ. 2) THEN
C*****************      PLOTTING THE PHASE PLANE      **************
```

```
            XAXL   = 4.0
            XOFF   = 0.29
            XNAME  = 'X1 STATE'
            XNAML  = -8
            YNAME  = 'X2 STATE'
            YNAML  =  8
            PNAME1 = 'X1 vs. X2 PHASE PLANE'
            PNAM1L = 21
            XORGN  = XORGN + 0.65
         ELSE
C*****************     PLOTTING THE TIME RESPONSE     **************
            XAXL   = 5.0
            XOFF   = 0.25
            XNAME  = 'REAL TIME (sec)'
            XNAML  = -15
            YNAME  = 'STATE TRAJECTORY'
            YNAML  =  16
         ENDIF
C
         YAXL   = 4.0
         ASPRAT = 0.70
         CHARHT = 0.23
         CHRHT2 = 0.8 * CHARHT
C*****************          PLOT TITLE LOCATIONS          ****************
         PTX1   = XOFF + (XAXL-PNAM1L*ASPRAT*CHARHT)/2.
         PTY1   = 4.74
         PTX2   = XOFF + (XAXL-PNAM2L*ASPRAT*CHRHT2)/2.
         PTY2   = 4.42
         PTX3   = XOFF + (XAXL-PNAM3L*ASPRAT*CHRHT2)/2.
         PTY3   = 4.1
         NCHAR1 = ifix(PNAM1L)
         NCHAR2 = ifix(PNAM2L)
         NCHAR3 = ifix(PNAM3L)
C
         CALL PLOTS(0,IOPORT,MODEL)
         CALL FACTOR(1.00)
         CALL ASPECT(ASPRAT)
C
         CALL SCALE(VY,YAXL,NPTS,1)
         IF (PLTYPE .EQ. 1) THEN
C          This scaling applies when the X axis represents DISCRETE TIME
           CALL SCALE (VTIME,XAXL,NPTS,1)
           CALL STAXIS(.15,.20,.12,.080,0)
         ELSEIF (PLTYPE .EQ. 2) THEN
C          This scaling applies when the X axis represents a STATE
           XLO = VTIME(1)
           XHI = VTIME(1)
           YLO =    VY(1)
           YHI =    VY(1)
           DO 15 J = 2,NPTS
                IF ( VTIME(J) .GT. XHI ) XHI = VTIME(J)
                IF ( VTIME(J) .LT. XLO ) XLO = VTIME(J)
                IF (    VY(J) .GT. YHI ) YHI =    VY(J)
                IF (    VY(J) .LT. YLO ) YLO =    VY(J)
  15       CONTINUE
           XRANGE = XHI - XLO
           YRANGE = YHI - YLO
           IF( YRANGE .LT. XRANGE ) THEN
               INCRMT         = XRANGE/XAXL
               VY(NPTS+1)     = YLO - ((YAXL*INCRMT - YRANGE)/2.)
               VTIME(NPTS+1) = XLO - INCRMT/2.
               INCRMT         = XRANGE/(XAXL-1.)
           ELSE
               INCRMT         = YRANGE/YAXL
               VTIME(NPTS+1) = XLO - ((XAXL*INCRMT - XRANGE)/2.)
               VY(NPTS+1)     = YLO - INCRMT/2.
               INCRMT         = YRANGE/(YAXL-1.)
           ENDIF
           VY(NPTS+2)        = INCRMT
           VTIME(NPTS+2)     = INCRMT
```

```
            CALL STAXIS(.15,.20,.12,.080,2)
         ELSE
C          This scaling applies when the X axis represents REAL TIME
            VTIME(NPTS+1) = BEGTIM
            VTIME(NPTS+2) = (VTIME(NPTS)-VTIME(NPTS+1))/XAXL
            CALL STAXIS(.15,.20,.12,.080,2)
         ENDIF
C
         FIRSTX    = VTIME(NPTS+1)
         DELTAX    = VTIME(NPTS+2)
         LASTX     = FIRSTX + DELTAX*XAXL
         FIRSTY    = VY(NPTS+1)
         DELTAY    = VY(NPTS+2)
         LASTY     = FIRSTY + DELTAY*YAXL
         IF (PLTYPE .EQ. 1  .OR. PLTYPE .EQ. 3) THEN
            VTIMSS(8) = BEGTIM
            VTIMSS(9) = (FINTIM - BEGTIM)/XAXL
         ELSE
            DO 20 J = 1,7
               VYSS(J)   = 0.0
               VTIMSS(J) = (((LASTX - FIRSTX)/6.) * (J-1) ) + FIRSTX
               VXXSS(J)  = 0.0
               VXYSS(J)  = (((LASTY - FIRSTY)/6.) * (J-1) ) + FIRSTY
   20       CONTINUE
            VTIMSS(8) = FIRSTX
            VTIMSS(9) = DELTAX
            VXXSS(8)  = FIRSTX
            VXXSS(9)  = DELTAX
            VXYSS(8)  = FIRSTY
            VXYSS(9)  = DELTAY
         ENDIF
         VYSS(8)   = FIRSTY
         VYSS(9)   = DELTAY
         CALL PLOT(XORGN,YORGN,-13)
         CALL PLOT(XAXL,0.0,3)
         CALL PLOT(XAXL,YAXL,2)
         CALL PLOT(0.00,YAXL,2)
         CALL AXIS(0.0,0.0,XNAME,XNAML,XAXL,0.,FIRSTX,DELTAX)
         CALL STAXIS(.15,.20,.12,.080,2)
         CALL AXIS(0.,0.,YNAME,YNAML,YAXL,90.,FIRSTY,DELTAY)
         CALL SYMBOL(PTX1,PTY1,CHARHT,PNAME1,0.,NCHAR1)
         CALL SYMBOL(PTX2,PTY2,CHRHT2,PNAME2,0.,NCHAR2)
         CALL SYMBOL(PTX3,PTY3,CHRHT2,PNAME3,0.,NCHAR3)
C
         CALL LINE(VTIME,VY,NPTS,1,0,0)
         IF( FIRSTY.LE.0 )THEN
            IF( LASTY.GE.0 )CALL CURVE(VTIMSS,VYSS,7,-0.1)
         ENDIF
         IF(PLTYPE .EQ. 2) THEN
            IF( FIRSTX.LE.0 )THEN
               IF( LASTX.GE.0 )CALL CURVE(VXXSS,VXYSS,7,-0.1)
            ENDIF
         ENDIF
         CALL PLOT(0.,0.,999)
C
         RETURN
         END
```

LIST OF REFERENCES

1. Dorf, R.C., *Modern Control Systems*, Addison-Wesley, 1983.

2. Meriam, J.L., *Engineering Mechanics Dynamics*, John Wiley and Sons, 1978.

3. Reid, J.G., *Linear System Fundamentals*, McGraw-Hill, 1983.

4. VanLandingham, H.F., *Introduction to Digital Control Systems*, Macmillan Publishing Co., 1985.

5. Astrom, K.J., Wittenmark, B., *Computer Controlled Systems Theory and Design*, Prentice-Hall, Inc., 1984.

6. Kirk, D.E., *Optimal Control Theory, An Introduction*, Prentice-Hall, Inc., 1970.

7. Athans, M., Falb, P.L., *Optimal Control, An Introduction to the Theory and its Applications*, McGraw-Hill, 1966.

8. Roskam, J., *Airplane Flight Dynamics and Automatic Controls*, Roskam Aviation and Engineering Corporation, 1982.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
4.	Professor Harold A. TITUS, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	10
5.	Professor Alex GERBA, Jr., Code 62Gz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	10
6.	Captain Scot D. LLOYD 3719 Frostwood Road Knoxville, TN 37921	5

Th
18767 3

SCHOOL

Thesis
L772 Lloyd
c.1 An autopilot design
for the United States
Marine Corps airborne
remotely operated device.

25 OCT 89 35216

Thesis
L772 Lloyd
c.1 An autopilot design
for the United States
Marine Corps airborne
remotely operated device.

thesL772
An autopilot design for the United State
3 2768 000 75438 6
DUDLEY KNOX LIBRARY